Игорь Котельников
Платон Чеботаев

LaTeX 2ε по-русски

Сибирский хронограф
Новосибирск
2004

ББК 32.98
УДК 681.322

Котельников И. А., Чеботаев П. З.
К26 LaTeX по-русски.— 3-е издание, перераб. и доп.— Новосибирск: Сибирский хронограф, 2004. — 496 с.: ил.
ISBN 5–87550–195–2

Книга представляет собой полное руководство по издательской системе LaTeX. Несравненная мощь этой системы наиболее зримо проявляется при подготовке научно-технических печатных изданий высшего качества, содержащих большое количество математических формул и иллюстраций. Книга построена как учебник с подробным описанием команд LaTeX'а и множеством примеров. В третьем издании отражены изменения в способе русификации системы LaTeX и добавлены сведения о преобразовании документов LaTeX'а в формат PDF.

Программное обеспечение, описанное в книге, распространяется преимущественно бесплатно и может быть загружено через интернет с общедоступных серверов CTAN или по адресу `http://www.tutornet.ru/tex/`. В части тиража к книге прилагается компакт-диск, где собрано всё необходимое для работы с системой LaTeX в операционных системах семейства Windows.

ISBN 5–87550–195–2

Целое больше, чем сумма частей.
Аристотель. Метафизика

Вместо предисловия

LaTeX — это настольная издательская система. Её применения простираются от подготовки одностраничных писем до создания многотомных фолиантов. Реализации LaTeX'а существуют для всех типов компьютеров. LaTeX упрощает работу с текстом, позволяя сосредоточить внимание на его содержании. Заботы по оформлению текста LaTeX принимает на себя. В исходном виде документ LaTeX является текстовым файлом и поэтому одинаково пригоден для компьютера в издательском офисе в Нью-Йорке, Мадриде или Новосибирске. Редакции научных журналов рекомендуют, а иногда и вынуждают готовить статьи в системе LaTeX и принимают их по электронной почте. Заменив всего лишь одно слово — название класса печатного документа в преамбуле входного файла, издатель придаст тексту тот облик, который отличает выбранный журнал и который при ином методе общения с издательством требует немалых затрат времени.

Книга, которую Читатель держит в руках, от первой до последней страницы сформатирована LaTeX'ом. Она сверстана на персональном компьютере, а затем размножена и сброшюрована в типографии.

Сначала был TeX

Программисты со стажем знают профессора Гарвардского университета Дональда Кнута (Knuth, Donald) как автора многотомной монографии «Искусство программирования для ЭВМ» [1]. Ровно 25 лет назад, в 1978 году, он опубликовал первую версию системы обработки печатных документов, известную ныне как TeX [2] и METAFONT [3]. Многие специалисты безоговорочно относят её к одному из выдающихся достижений XX столетия, приравнивая к созданию печатного станка Гутенбергом (Gutenberg, Iohann). TeX предвосхитил идеи, получившие признание на рубеже третьего тысячелетия. Система команд TeX по сути была первым языком разметки гипертекстов, к которым принадлежит широко известный ныне HTML (Hyper Text Markup Language) — язык разметки документов для интернета. Исполняемая программа `tex`, выполняющая преобразование размеченного текста в документ, пригодный для высококачественной печати, была чуть ли не первой из программ, которые сейчас принято называть парсерами (parser).

TeX общепризнанно считается наиболее качественной системой подготовки печатных публикаций. Как сказано в словаре компьютерных терминов [4], TeX определяет стандарт, к которому пытаются приблизиться другие настольные издательские системы.

Затем пришел LaTeX

Следующий шаг сделал Лесли Лампорт (Lamport, Leslie). В начале восьмидесятых годов XX века он разработал систему подготовки печатных документов LaTeX [5], основанную на форматирующих средствах TeX'а. LaTeX позволил пользователю сконцентрировать свои усилия на содержании и структуре текста, не заботясь о деталях его оформления. Как и профессор Кнут, Л. Лампорт опередил своё время. Идея отделения содержания от формы, реализованная в системе LaTeX, нашла своё продолжение в XML — расширяемом языке разметки (eXtensible Markup Language), появившемся в конце девяностых годов XX века. Простая замена *стиля* документа[1] в системе LaTeX, как и замена стиля XSL (eXtensible Style Language), «надеваемого» на разметку XML, способна одинаково радикально изменить внешний вид документа.

LaTeX реализован в виде *формата*, то есть надстройки над компактной системой базовых *команд*, встроенных в исполняемую программу `tex`. Надстройка, созданная самим Кнутом, называется «формат Plain TeX», или просто TeX. Формат TeX входит составной частью в формат LaTeX.

LaTeX содержит удобные средства генерации алфавитного указателя, списков литературы, рисунков и таблиц, развитые средства импортирования графики, обеспечивает автоматическую нумерацию формул, ссылок и других подобных объектов в сочетании с эффективным механизмом перекрёстного цитирования. Подлинного совершенства TeX и LaTeX достигли в форматировании математических формул. Ни одна другая издательская система не сумела достичь тех же вершин в этой области издательского ремесла. Поэтому LaTeX особенно популярен в научных кругах. За два десятилетия после изобретений Д. Кнута и Л. Лампорта появились прекрасные текстовые процессоры, но TeX и LaTeX сохраняют ранее завоёванные позиции. Причина очевидна: уникальное качество подготовки печатной продукции, помноженное на полную совместимость версий TeX'а и LaTeX'а для различных компьютеров.

В конце восьмидесятых годов TeX и LaTeX достигли России. Был разработан алгоритм автоматического переноса русских слов. Кириллические шрифты разрабатывались в разных местах: в Вашингтонском университете, в Институте высоких энергий в Протвино, в издательстве «Мир».

Время ε

Формат LaTeX является открытым, так как переносится с компьютера на компьютер в текстовом виде, а Д. Кнут даже подробно описал «внутреннее устройство» формата Plain TeX в своей книге [2]. Неудивительно поэтому, что вслед за Plain TeX и LaTeX были разработаны другие форматы. Наибольшую популярность приобрели SliTeX всё того же Лесли Лампорта (Lamport, Leslie) [5] и AMS-TeX Майкла Спивака (Spivak, Michael) [6]. Формат SliTeX ориентирован

[1] Сейчас *стиль* документа LaTeX принято называть *классом*.

на подготовку слайдов, которые печатаются на листах прозрачной плёнки и используются для проекции на экран при выступлениях перед большой аудиторией. AMS-TeX, разработанный по заказу Американского математического общества, предназначен для обработки особо изощрённых научных текстов.

Интернационализация круга пользователей породила множество диалектов популярных форматов. В этих условиях Д. Кнут объявил, что замораживает TeX в виде, который тот приобрёл к началу 90-х годов, одновременно раскрыв исходные коды исполняемых программ своей системы.

Примерно в это же время Франк Миттельбах (Mittelbach, Frank), Крис Роули (Rowley, Chris) и Райнер Шопф (Schöpf, Rainer) объявили о начале работы над проектом LaTeX3. Целью проекта было создание формата LaTeX3, который исключил бы множественность форматов, но позволил расширять функциональные свойства путём подключения дополнительных *пакетов*.

Результатом этой работы, первоначально названным промежуточным, стал выпуск в 1993 году версии, получившей название LaTeX 2_ε. В этой версии все три формата: LaTeX, SliTeX и AMS-TeX — объединены в один. Он сохранил название LaTeX. Миграция от предыдущей версии 2.09 была максимально облегчена, так как LaTeX 2_ε имеет режим эмуляции, который позволяет работать со старыми документами LaTeX 2.09. SliTeX стал одним из *классов* в новом формате, а AMS-TeX разбит на несколько классов и пакетов. В LaTeX 2_ε унифицирована работа с графикой, цветом и шрифтами. Наконец, LaTeX 2_ε стал воистину интернациональной системой подготовки печатных документов после того, как в 1999 году была завершена стандартизация кодировки шрифтов для большинства языков народов мира, включая славянские языки. Однако за прошедшие годы LaTeX 2_ε так и не вырос в LaTeX3. Время ε продолжается.

PDFLaTeX

При видимой стабильности языка разметки LaTeX все последние годы происходила незримая работа по совершенствованию исполняемых модулей системы LaTeX. Малозаметные улучшения, столь многочисленные, что их просто невозможно здесь перечислить, переросли в новое качество в начале XXI века. Русские пользователи LaTeX'а могут вести отсчёт новой эры с ноября 2001 года, когда достоянием общественности стали шрифты cm-super, разработанные Владимиром Воловичем[2]. Их появление фактически сделало ненужным традиционный сценарий компиляции исходного текста с разметкой LaTeX в `dvi`-файл, так как теперь есть все необходимое для прямого преобразования размеченного текста в `pdf`-файл. Если для преобразования исходного текста с разметкой LaTeX в формат DVI (DeVice Independent), который был разработан Д. Кнутом специально для системы TeX, нужно использовать программу `latex`, то для преобразования того же текста в формат PDF (Portable Document Format), который в настоящее время

[2] С середины 2002 года эти шрифты вошли в библиотеку программ MiKTeX для операционной системы Windows. Их можно установить при помощи MiKTeX Package Manager.

доминирует в электронном документообороте, нужно использовать программу `pdflatex`. Строго говоря, и раньше документы LaTeX можно было преобразовать в формат PDF, но при этом либо происходило ухудшение качества изображения, либо в исходный текст необходимо было вносить определённые изменения (например, подменять шрифты и рисунки), либо выполнять преобразование в 2–3 шага. Теперь в этом нет необходимости.

Далее следует...

Признаемся, было время, когда нам казалось, что LaTeX обречён на скорое угасание. Не потому, что у него вдруг объявились сильные конкуренты, а из-за внутренних ограничений, ставших неактуальными вследствие стремительного развития компьютерной техники. Например, когда-то генерация комплекта растровых изображений шрифтов METAFONT занимала много часов. Именно поэтому растровые шрифты приходилось делать заранее. Современные системы выполняют растеризацию «на лету», так что новое поколение пользователей может и не знать, что такое растровые шрифты. А ведь только растр, то есть набор точек разного цвета, могут наносить на бумагу принтеры любого типа, от струйного до лазерного. Точно так же изображение на экране любого монитора является растровым, то есть состоит из точек. Попытки преодолеть атавизмы TeX'а предпринимались и предпринимаются по сей день, хотя альтернативные системы, такие как Omega [7], так и не достигли уровня популярности старого доброго TeX'а.

Растущая популярность интернета и вызванная им потребность в обмене информацией между различными компьютерными платформами породили концепцию разделения содержания и формы, предвестником которой, как мы отмечали, был LaTeX. Новейшие Web обозреватели (программы для навигации по интернету) уже умеют качественно отображать математические формулы, записанные на языке разметки математических формул MathML [9]. Он имеет много общего с языком LaTeX. Поэтому следует ожидать появления программных продуктов, которые в равной степени будут способны сохранять подготовленные тексты в разметке как LaTeX, так и MathML. И хотя неизбежно придёт то время, когда LaTeX уступит место более совершенным издательским системам, сейчас ясно, что LaTeX — это надолго.

Логическое против визуального

Люди, впервые знакомящиеся с LaTeX'ом, обычно говорят, что очень неудобно не видеть сразу, как будет выглядеть отпечатанная страница. Не будем спорить — они правы. Однако после пары уроков самый недоверчивый Читатель обнаружит, что зримо представляет себе результат своих действий так же, как гроссмейстер предвидит исход шахматных комбинаций на много ходов вперёд.

В текстовых процессорах, которые позволяют в момент набора текста увидеть его на экране дисплея точно таким, как он будет выглядеть на бумаге, исповедуется концепция *визуального проектирования* (сокращённо WYSIWYG от английских слов What You See Is What You Get — «что видите, то и получите»). LATEX создан для *логического проектирования* печатного документа, которое позволяет сосредоточиться на содержании текста, возлагая на компьютер заботу по его оформлению. Используя знания самых опытных типографских дизайнеров, LATEX справится с этой задачей лучше автора текста, который не обязан быть знатоком типографского дела. Впрочем, если нужно написать деловое письмо, а клавиатура компьютера пугает количеством кнопок, отложите в сторону эту книгу и начните работать с Microsoft Word. Это чудный программный монстр. Он заставит компьютер трудиться в полную силу, и очень скоро обнаружится, что коротенький текст почему-то занимает мегабайт (или десять) дисковой памяти. Если же Вы регулярно готовите сложные тексты с большим количеством математических формул, таблиц и рисунков, то LATEX станет Вашим надёжным помощником, а наша книга — учебником и справочником.

Преимущество логического проектирования над визуальным состоит в его гибкости. При визуальном проектировании «что видите, *только* то и получите». Текст, однажды подготовленный к печати, при таком подходе иной раз легче переписать заново, чем поправить. Представьте, например, что нумерацию всех формул нужно перенести с правой стороны страницы на левую. Между тем LATEX справится с этой задачей играючи — достаточно лишь указать соответствующую *опцию* при выборе *класса* печатного документа.

В наши дни то, что Л. Лампорт назвал логическим проектированием, связывают с концепцией разделения содержания и формы. Идеи логического проектирования проникли даже в визуальные редакторы, такие как Microsoft Word. Однако лишь немногие пользователи таких редакторов умеют использовать преимущества управления стилем документа, предпочитая «раскрашивать каждую букву по отдельности». Верно и обратное: логическое проектирование испытывает немалое влияние со стороны визуального. Существуют полувизуальные редакторы, например Scientific Word. В них на стадии разметки текста можно конструировать формулы, таблицы и т. д. из визуальных заготовок, которые перед компиляцией документа переводятся в команды, напоминающие команды LATEX'а, а затем запускается компилятор наподобие LATEX'а.

(LA)TEX

LATEX исповедует более современную концепцию разделения содержания и формы. TEX более тяготеет к визуальному проектированию. Его можно отнести к языкам программирования более низкого уровня, нежели LATEX. В наши дни TEX используется главным образом при разработке классов и пакетов LATEX'а. Для подготовки печатных документов лучше подходит LATEX.

Так как LaTeX написан в командах TeX'а, в тексте, предназначенном для LaTeX'а, с некоторой осторожностью можно использовать почти любые команды TeX'а. Строго говоря, указать границу, разделяющую TeX и LaTeX, не так-то просто. Например, LaTeX и TeX используют одинаковые команды набора математических символов, а их очень и очень много. Впрочем, можно руководствоваться простым правилом: все команды, описанные в руководстве Л. Лампорта [5] или в нашей книге, являются командами LaTeX'а. Мы старались без необходимости не углубляться в дебри TeX'а, чем грешат многие современные руководства по LaTeX'у. LaTeX — в том виде, как его придумал Л. Лампорт,— был очень компактной, тщательно продуманной системой. Это и сделало его популярным. И очень жаль, что развитие LaTeX'а отчасти пошло в противоположном направлении. Большинству пользователей некогда читать многотомные талмуды. Часто важнее быстро получить качественно оформленный печатный документ. Если в нашей книге и упоминается слово TeX, то обычно в качестве сокращения от сочетания «TeX и LaTeX» или если нужно подчеркнуть общее происхождение этих двух издательских систем.

С чего начать

Для начала условимся, как произносить слова TeX и LaTeX. Отец TeX'а Дональд Кнут утверждает [2, стр. 1], что по-русски надо говорить *тех*, а Лесли Лампорт называет своё дитя [5] именем *латех*, хотя считает, что *латекс*[3] тоже звучит неплохо. Такова родительская воля.

LaTeX появился в те времена, когда вычислительные машины повсеместно были объектами коллективного пользования, но он был и остаётся инструментом индивидуального назначения. Следовательно, LaTeX идеально подходит для персонального компьютера.

Всё необходимое программное обеспечение и национальные шрифты можно бесплатно загрузить через интернет из общедоступных серверов. Перечислим основные адреса:

`ftp://ftp.dante.de/tex-archive/`	(Heidelberg, Germany)
`http://www.tex.ac.uk/tex-archive/`	(Cambridge, U.K.)
`ftp://ctan.tug.org/tex-archive/`	(Vermont, U.S.A.)
`ftp://ftp.radio-msu.net/pub/tex/`	(Москва)
`ftp://ftp.chg.ru/pub/TeX/CTAN/`	(С.-Петербург)

Имеются также коммерческие версии:

`http://www.yandy.com/`	(pcTeX)
`http://www.pctex.com/`	(Personal TeX)
`http://www.tcisoft.com/`	(Scientific Word)
`http://www.micropress-inc.com/`	(VTeX)

[3] В переводе с латыни «латекс» означает «млечный сок». Латексом также называют материал, используемый в строительстве.

Дополнительные адреса поставщиков программного обеспечения для издательской системы LaTeX и много другой полезной информации можно найти на сайте организации TUG (TeX Users Group), созданной для популяризации TeX'а:

`http://www.tug.org/`

Она поддерживает сеть общедоступных хранилищ информации CTAN (Comprehensive TeX Archive Network). Информацию о CTAN можно найти в интернете по адресу

`http://www.ctan.org/`

В части тиража к нашей книге прилагается компакт-диск, где собрано программное обеспечение для работы с LaTeX'ом на платформе Windows. Диск содержит бесплатные (freeware) и условно бесплатные (shareware) программные продукты. Постоянно обновляемая версия копии компакт-диска размещена в интернете по адресу

`http://www.tutornet.ru/tex/`

Документация, поставляемая в составе программного обеспечения, объясняет, как установить и запустить в работу LaTeX на компьютере нашего Читателя. При установке программного обеспечения с прилагаемого компакт-диска следует начать с инструкции `readme.htm`. Работая над книгой, мы пользовались программным обеспечением только с этого компакт-диска. По большому счёту оно состоит из трёх частей.

Первая часть распространяется в виде текстовых файлов, из которых затем генерируется *формат* LaTeX и все необходимые служебные файлы. Эти файлы содержат полное описание LaTeX'а, включая его «внутреннее устройство». Процедура генерации формата начинается с удаления из исходных файлов всех комментариев и выполняется при помощи программы `latex`, той самой, которая используется для компиляции документов LaTeX. Обычно генерация формата выполняется автоматически мастером установки издательской системы LaTeX.

Вторая часть — исполняемая программа `latex` — индивидуальна для каждой компьютерной платформы (операционной системы). Эта программа может называться `tex.exe`, `latex.exe`, `pdflatex.exe` или как-то иначе. На самом деле, требуется не одна программа, а целый набор, включающий программы для генерации шрифтов, для вывода готового печатного документа на принтер и для его просмотра на экране. Бо́льшая часть таких программ берет начало от кодов, открытых для общего пользования Дональдом Кнутом (Knuth, Donald). Значителен вклад и других авторов. Например, программа `dvips` написана Томашем Рокицки (Rokicki, Tomas) и предназначена для вывода документа на PostScript-принтер. Программа *MakeIndex* Пехона Чженя (Chen, Pehong) используется для сортировки алфавитного указателя. Программа BibTeX Орена Паташника (Patashnik, Oren) предназначается для работы с библиографией. Все такие программы поставляются в составе *реализации* системы LaTeX, приспособленной к определённой платформе.

В среде Windows наибольшее число приверженцев завоевала библиотека программ MiKTeX Кристиана Ченка (Schenk, Christian). Её можно загрузить с сайта

`http://www.miktex.org/`

Библиотека MiKTeX включает огромное количество пакетов, позволяет загружать отсутствующие пакеты из интернета. Мастер обновления MiKTeX помогает оперативно скачивать через интернет обновлённые пакеты и исполняемые модули. Альтернативой MiKTeX является библиотека программ fpTeX:

`http://www.tug.org/fptex/`

Она ведёт свою родословную от системы teTeX Томаса Эссера (Esser, Thomas), которая популярна среди приверженцев различных вариантов операционной системы Unix:

`http://www.tug.org/tetex/`

Для компьютеров Macintosh существует очень качественная библиотека программ OzTeX Эндрю Треворова (Trevorrow, Andrew):

`http://www.trevorrow.com/oztex/`

Можно также рекомендовать библиотеку TeXShop Ричарда Коха (Koch, Richard):

`http://www.uoregon.edu/~koch/texshop/`

Она интересна последовательной реализацией идеи перехода от DVI к формату PDF, который является «родным» для операционной системы Mac OS, начиная с версии Mac OS X.

Неотъемлемым элементом любой современной реализации издательской системы LaTeX являются программы для работы с графикой PostScript. Часто они поставляются отдельно от системы LaTeX. Например, MiKTeX, fpTeX и teTeX используют для своих целей библиотеку Ghostscript:

`http://www.cs.wisc.edu/~ghost/`

Для неё разработан графический интерфейс GSview, который позволяет использовать эту библиотеку в качестве самостоятельного продукта:

`http://www.cs.wisc.edu/~ghost/gsview/`

Третью составную часть чаще всего называют редактором, потому что она используется для редактирования документов LaTeX. Редактор также выполняет функции диспетчера, организующего взаимную работу программ из библиотеки MiKTeX или ей подобной. Наличие специализированного редактора вовсе не обязательно, так как исходный текст документа LaTeX можно редактировать в любом текстовом редакторе, например в блокноте `notepad`, который есть в любой версии Windows. Также необязательно иметь программу-диспетчера, так как

программы MiKTeX можно запускать из командной строки (но в Windows эту строку ещё надо найти).

Лучшим на сегодняшний день специализированным редактором документов LaTeX для Windows является WinEdt. Ранее он распространялся бесплатно, но начиная с версии 5 перешел в разряд условно бесплатных (shareware) программ. После месяца работы с WinEdt требуется платная регистрация, но вместо него можно использовать другие редакторы для подготовки документов, такие как `TeXnicCenter`, который распространяется бесплатно и также имеется на прилагаемом компакт-диске.

Доподлинно неизвестно, какова доля пользователей LaTeX'а, работающих на платформах Unix и Mac. Известно, что таких пользователей немало. Многие программы, используемые системой LaTeX, несут на себе явный отпечаток «юниксоидности» их авторов. Но мы, авторы этой книги, как и многие наши коллеги, работаем в среде Windows. Поэтому мы давно выбрали связку `WinEdt`+MiKTeX. Это не могло не наложить отпечаток на нашу книгу. В данном, третьем издании мы решили отойти от табу предыдущих двух изданий, вышедших в 1994 и 1998 годах в издательстве «Сибирский хронограф», и не скрывать более наших пристрастий. По просьбам читателей мы отважились немного рассказать о том, как работать с программами MiKTeX. Иногда без подобных сведений просто невозможно обойтись. Например, мы расскажем, как заставить BibTeX сортировать библиографический указатель по русскому алфавиту или как расположить страницы в выходном файле в порядке, пригодном для изготовления буклета. Мы рассчитываем, что этот рассказ будет полезен пользователям других реализаций LaTeX, так как все они происходят от общих корней.

Однако основное содержание книги по-прежнему составляет LaTeX как *язык программирования*, то есть первая составная часть издательской системы LaTeX.

Как читать эту книгу

Первая глава является вводной. Ознакомившись с её содержанием, Читатель сможет готовить к печати несложные тексты с высоким качеством дизайна. Мы рекомендуем ознакомиться с примером исходного текста документа LaTeX из раздела 1.3 и сразу начинать набирать свой текст, обращаясь к оглавлению для поиска нужных разделов книги. Для поиска нужных команд можно использовать алфавитный указатель в конце книги.

Вторая глава предназначена тем, кто уже приобрёл некоторый опыт работы с LaTeX'ом. Она описывает общие правила синтаксиса LaTeX'а и объясняет такие фундаментальные понятия, как *класс*, *пакет*, *команда*, *декларация*, *процедура*, *счётчик*. Читатель, только начинающий знакомство с LaTeX'ом, может смело пропустить при первом чтении эту главу, так как большинство используемых далее терминов интуитивно понятны.

В третьей и последующих главах на большом числе примеров объясняются различные аспекты подготовки текстов: набор формул, оформление таблиц, ри-

сунков и т. д. Эти главы можно изучать по мере необходимости и почти в любом порядке, хотя их содержание и не является взаимно независимым. Точно так же маленький ребёнок учится читать, ещё не зная правил правописания. Однако браться сразу же за последние главы не следует, так как ближе к концу книги изложение становится более конспективным.

Глава 17 предназначена самым опытным пользователям, которые уже подготовили к печати целую книгу и вступили в стадию переговоров с издательством о деталях её оформления, о том, что называется макетом полосы набора. В главе 18 рассказано, как подготовить электронную версию книги для распространения через интернет.

Мы старались придерживаться такого правила: сначала для каждой команды или процедуры привести её определение в наиболее общем виде, а затем описать её назначение и пояснить её действие на примерах.

Определения заключены в рамку:

```
\documentclass[options]{class}[release-date]
```

В предметном указателе номера страниц, где находятся такие определения, выделены *курсивом*. Небольшая часть команд имеет несколько определений. Такое обычно случается, если команда изменена пакетом. Название пакета (или класса) указывается в скобках справа от рамки:

```
\begin{verbatim}  ...  \end{verbatim}
```

(verbatim)

Определение некоторых команд снабжено значком ⚠. Его смысл разъясняется в разделе 2.7 второй главы.

Примеры по бо́льшей части сформатированы в две колонки:

```
левая колонка содержит исходный
текст с командами \LaTeX'a, а
правая~--- этот же текст в
<<напечатанном>> виде.
```

левая колонка содержит исходный текст с командами LaTeX'а, а правая — этот же текст в «напечатанном» виде.

Исходный текст в примерах набран специальным `машинописным шрифтом`, чтобы подчеркнуть, что он вводится с клавиатуры компьютера. В определении команды машинописный шрифт означает, что соответствующую часть команды нельзя варьировать. Напротив, текст, набранный прямым курсивом, можно заменять другим. Некоторые обозначения в зависимости от контекста могут быть набраны в разном регистре или разными шрифтами. Например, PDF обозначает формат документа, а `pdf` — расширение имени файла, содержащего документ. Логотип LaTeX употребляется, когда речь идет об издательской системе в целом или имеется в виду какой-то элемент этой системы, но `latex` всегда обозначает исполняемую программу.

Материал, носящий заведомо справочный характер, набран таким же шрифтом, как данный абзац. При беглом чтении эти сведения можно пропустить.

Послесловие к предисловию

Наконец, немного о том, как мы писали эту книгу.

В 1994 году в новосибирском издательстве «Сибирский хронограф» вышло её первое издание [10], благосклонно встреченное читателями. Оно имело другое название, а один из авторов скрылся под псевдонимом, но мы ведём счёт именно от этой книги. Она содержала наиболее полное из опубликованных на то время на русском языке описаний версии LaTeX 2.09 и очень скоро стала библиографической редкостью. Примерно в то же время появилась новая версия LaTeX 2_ε, и нам неоднократно предлагали переиздать книгу, но мы решили, что не будем этого делать, пока не отразим в ней произошедших изменений. Работа заняла 4 года, так как мы не пишем о том, чего не проверили сами. Второе издание [11] вышло 1998 году в том же издательстве. Нам было приятно увидеть наш труд в одном из рейтингов на втором месте, особенно почётном оттого, что первую строчку занимала книга Дональда Кнута [2].

И вот ещё почти через 6 лет выходит новое издание. Мы теперь живем в разных городах, а вектор наших интересов заметно поменял направление. LaTeX для нас уже не объект исследования, а инструмент подготовки очередной научной статьи.

В новом издании мы попытались отразить изменение нашего подхода к LaTeX'у в строну большего прагматизма. Например, при наличии нескольких пакетов, выполняющих примерно одинаковые функции, мы стремились выбрать 1–2, не утруждая Читателя сравнением всех альтернатив. Сделать выбор часто составляет самую сложную часть задачи. Именно эту часть мы постарались решить, отобрав самое необходимое, и восстановить тем самым идею, заложенную в основание LaTeX'а Лесли Лампортом. Читатель, не удовлетворённый таким подходом, может обратиться к серии переводных книг «Библиотека издательских технологий», выпускаемой издательством «Мир» [12–14].

В 3-м издании мы кое-что сократили, добавили несколько параграфов и главу о том, как подготовить документ LaTeX для показа в интернете. Но наибольшие переделки были вызваны изменением способа русификации LaTeX'а, которое произошло вскоре после выхода 2-го издания. В этой связи нам пришлось переписать добрую половину глав. И хотя не все технические решения, принятые группой разработчиков LaTeX'а, совпали с тем, что мы пропагандировали, мы безжалостно изъяли из текста книги всякие уклонения от «генеральной линии».

И. А. Котельников, П. З. Чеботаев

Новосибирск — Геленджик
30 апреля 2004 г.

Эта штука работает лучше,
если её включить.
Закон Сэттингера

Глава 1

Пособие для начинающих

Взамен авторучки и пишущей машинки LaTeX предлагает набор инструментов. Это *команды* и *процедуры* LaTeX'а. С наиболее важными из них знакомит настоящая глава. Изучив её, Читатель может смело браться за подготовку своего первого печатного документа. Мы советуем так и сделать, отложив дальнейшее чтение до той поры, когда возникнет понимание неполноты собственных знаний.

1.1. Входной файл

LaTeX преобразует размеченный *исходный текст* в *печатный документ*. Следуя новомодной терминологии, процесс преобразования нужно было бы назвать *парсингом*. Английский глагол to parse означает «делать грамматический разбор». Так что термин *парсинг* чрезвычайно точно отражает суть процесса. Однако Д. Кнут называл программу, выполняющую преобразование, *компилятором*. Он писал: «This is TeX, a document compiler intended to produce typesetting of high quality»[1].

Исходный текст и печатный документ — это то, что в докомпьютерную эпоху соответственно называлось рукописью и типографским оттиском. Помимо собственно «рукописи» LaTeX должен получить указания, что с ней делать в виде *разметки*. Размеченный исходный текст записывается во *входной файл*, который может быть создан с помощью любого *редактора*, способного сохранять файлы в *текстовом формате*. Многие редакторы записывают файл в своём собственном формате, непонятном другим текстовым процессорам, однако они обычно могут *экспортировать* его в текстовый формат. Там, где это не вызовет недоразумений, мы будем использовать термины *исходный текст* и *входной файл* как синонимы.

Имя файла jobname.ext состоит из двух частей: jobname собственно и есть *имя файла*, а ext — *расширение* имени файла. Входной файл для LaTeX'а, как правило, имеет расширение `tex`. Точно такое же расширение имеют входные файлы, содержащие исходный текст с разметкой для Plain TeX'а, поэтому многие пользователи LaTeX'а дают своим исходным файлам расширение `ltx`. Однако в

[1] Это TeX, компилятор для подготовки печатных документов высокого качества.

нашей книге мы не будем нарушать традицию, предполагая, что везде, где специально не оговорено иное, исходный текст с разметкой LaTeX записан в файл с расширением `tex`.

В самом начале входной файл для LaTeX'а должен содержать команду[2]

\documentclass[*options*]{*class*}[3]

в которой [*options*] и {*class*} являются соответственно необязательным и обязательным аргументами. Обязательный аргумент в фигурных скобках должен содержать название *класса печатного документа*. Существует 6 стандартных классов: `article`, `letter`, `report`, `book`, `proc`, `slides`, которые имеются в самом минимальном варианте издательской системы LaTeX. Это просто текстовые файлы с расширением `cls`. Кроме того, издатели журналов разработали для своих специфических целей множество нестандартных классов. Необязательный аргумент вместе с указывающими на его необязательность квадратными скобками может вообще отсутствовать. В необязательном аргументе может присутствовать любое количество *опций*, разделённых запятыми. Опции модифицируют *стиль* (способ оформления) печатного документа, определяемый выбором его класса. Следующей обязательной командой является

```
\begin{document}
```

Текст перед `\begin{document}` называется *преамбулой*. Преамбула обычно содержит команды, производящие дополнительную настройку выбранного класса печатного документа, а также определения новых команд LaTeX'а. Собственно текст документа начинается после `\begin{document}`, а заканчивается командой

```
\end{document}
```

Всё, что следует за `\end{document}`, LaTeX попросту игнорирует. Перечисленные три команды дают основные указания компилятору, как должен выглядеть печатный документ. Если одна из них пропущена, LaTeX выдаст сообщение об ошибке при компиляции входного файла.

1.2. Кое-что о классе документа

Первая же команда во входном файле заставляет задуматься, к какому *классу* должен принадлежать подготавливаемый печатный документ. С некоторой долей лукавства можно утверждать, что для начала можно выбрать любой из шести стандартных классов. Дело в том, что команды, специфичные для каждого класса, легко пересчитать по пальцам. В остальном различие между классами сводится к размеру шрифтов в заголовках и способу нумерации глав, рисунков и таблиц.

---
[2] Ей могут предшествовать только строки комментария, начинающиеся с символа %.
[3] Здесь и далее мы опускаем знаки препинания, следующие за примерами команд.

Класс article (статья) лучше всего подходит для коротких текстов, но может удовлетворить большинство других потребностей. Он наиболее универсален, хотя техническому отчёту, возможно, более соответствует класс report (отчёт), а название класса book (книга) говорит само за себя. Если предстоит написать десяток писем нескольким адресатам, следует выбрать класс letter (письмо). Класс proc (доклад) предназначен для научных публикаций в трудах конференций. Класс slides (слайды) используют для подготовки демонстрационных материалов, набранных крупным шрифтом.

Полная информация о стандартных классах и доступных опциях содержится в главе 3. Специфические особенности отдельных классов рассмотрены в главе 15. Там же дано подробное описание класса revtex4 в качестве примера одного из самых популярных нестандартных классов.

1.3. Пример входного файла

Лучший способ освоить LaTeX — одолжить у знакомого «TeXсперта» какой-нибудь свободный от ошибок входной файл и начать мелкими порциями вставлять в него свой текст. Можно также взять один из учебных файлов small2e.tex или sample2e.tex, поставляемых в составе любой реализации системы LaTeX, или подготовить с помощью текстового редактора файл примерно следующего содержания (номера строк вводить не следует, они приведены исключительно для удобства Читателя).

```
% Символ % указывает, что текст, следующий за ним до конца
% строки, игнорируется и может использоваться в качестве комментария.

\documentclass{article}      % Класс печатного документа.
                             %
\usepackage[cp1251]{inputenc}% Кодировка исходного текста.
\usepackage[russian]{babel}  % Поддержка русского языка.
\usepackage{indentfirst}     % Отступ в первом абзаце.
                             %
\title{Образец текста}       % Заголовок документа.
\author{Н.\,Е. Образцов}     % Автор документа.
\date{8 июня 2002 года }     % Используйте \date{\today},
                             %  чтобы напечатать текущую дату.
\begin{document}             % Конец преамбулы, начало текста.

\maketitle                   % Печатает заголовок, список авторов и дату.

\begin{abstract}             % Печатает аннотацию.
   Это образец входного файла. Сравнивая его с готовым печатным
   документом, нетрудно освоить азы работы с
   \LaTeX'ом.                % Команда \LaTeX печатает логос.
```

```
\end{abstract}

\section{Обычный текст}      % Печатает заголовок раздела.
                             % Заголовки подразделов печатают
                             %  аналогичные команды
                             %  \subsection и \subsubsection.
Окончания слов и предложений отмечаются, как обычно,
пробелами.  Не имеет значения, сколько пробелов Вы
наберёте;  один пробел так же хорош, как    и    100.


Одна или несколько пустых строк обозначают конец абзаца.

Поскольку любое количество пробелов рассматривается как один,
способ форматирования текста во входном файле безразличен для
      \LaTeX'a.
Однако разумное форматирование входного файла облегчает его
чтение, проверку и внесение изменений.

\subsection{Математические выражения}
\LaTeX\ превосходно печатает как простые математические
уравнения типа
   \( x-3y = 7 \),
так и более сложные.
Математическую формулу можно записать отдельной
строкой:
   \[  x' + y^{2} = z_{i}^{2}. \]
Чтобы пронумеровать формулу, используйте процедуру \texttt{equation}:
\begin{equation}
   \int_{-\infty}^{\infty} dx \exp(-x^2) = \sqrt{\pi}.
\end{equation}
\begin{center}               % Центрирует текст.
   \Large                    % Команда \Large переключает
                             %  размер шрифта на больший.
   Всё остальное Вы узнаете,\\ прочитав эту книгу.
\end{center}
\end{document}               % Конец текста.
```

Пусть этот файл называется `first.tex`. Запустите компилятор `latex`. Если на компьютере установлен пакет программ MiKTeX, как мы предполагаем далее, это можно сделать из командной строки Windows:

```
latex first.tex
```

Расширение имени входного файла `.tex`, включая точку, можно опустить. По окончании работы программы `latex` появится файл с именем `first.dvi`, содер-

жащий документ формата DVI. Его можно просмотреть на экране дисплея, выполнив команду

```
yap first.dvi
```

причём расширение имени файла `.dvi` вновь можно опустить, поскольку именно оно предполагается программой `YAP` *по умолчанию*. Название программы `YAP` расшифровывается как Yet Another Previewer[4]. С недавних пор программы просмотра документов DVI всё чаще называют обозревателями или браузерами, следуя терминологии, пришедшей из интернета. Ранее такие программы называли превьюверами. Существует много разных DVI-обозревателей, `YAP` — один из самых поздних по времени появления. Отсюда и такое необычное название. Мы приводим вид командной строки главным образом для тех читателей, кто желает понять, каким образом редактор-диспетчер типа WinEdt организует взаимодействие десятков исполняемых программ, составляющих систему LaTeX. Работая с редактором, совмещающим в себе функции диспетчера LaTeX'а, проще запускать все эти программы через меню.

То, что Читатель увидит в окне DVI-обозревателя, должно выглядеть примерно так, как на рис. 1.1. Теперь можно попробовать увеличить линейные размеры документа примерно на 20%, добавив опцию `12pt` в первую команду:

```
\documentclass[12pt]{article}
```

и повторив компиляцию документа. Не менее просто отформатировать документ в две колонки. Для этого достаточно добавить ещё одну опцию в необязательный аргумент команды `\documentclass`:

```
\documentclass[12pt,twocolumn]{article}
```

Опции следует перечислять через запятую. Подробнее о форматировании документа в две или более колонки мы расскажем в главе 17.

Распечатать полученный документ на бумаге можно непосредственно из DVI-обозревателя. Однако для распространения в электронном виде документ DVI не очень подходит. Если просто отправить `first.dvi` по электронной почте, получатель сможет увидеть и распечатать документ только в том случае, если у него установлена система LaTeX. А если документ содержит рисунки или какие-то редкие шрифты, то их придётся посылать отдельно, так как компилятор `latex` *не внедряет* рисунки и шрифты в `dvi`-файл; и шрифты, и рисунки загружает DVI-обозреватель одновременно с загрузкой `dvi`-файла. К счастью, документ LaTeX нетрудно преобразовать в формат PDF, специально разработанный фирмой Adobe для электронного документооборота. Проще всего это сделать, заменив компилятор `latex` на `pdflatex`:

```
pdflatex first.tex
```

[4] Ещё один превьювер.

Образец текста

Н. Е. Образцов

8 июня 2002 года

Аннотация

Это образец входного файла. Сравнивая его с готовым печатным документом, нетрудно освоить азы работы с LaTeX'ом.

1 Обычный текст

Окончания слов и предложений отмечаются, как обычно, пробелами. Не имеет значения, сколько пробелов Вы наберёте; один пробел так же хорош, как и 100.

Одна или несколько пустых строк обозначают конец абзаца.

Поскольку любое количество пробелов рассматривается как один, способ форматирования текста во входном файле безразличен для LaTeX'а. Однако разумное форматирование входного файла облегчает его чтение, проверку и внесение изменений.

1.1 Математические выражения

LaTeX превосходно печатает как простые математические уравнения типа $x - 3y = 7$, так и более сложные. Математическую формулу можно записать отдельной строкой:

$$x' + y^2 = z_i^2.$$

Чтобы пронумеровать формулу, используйте процедуру `equation`:

$$\int_{-\infty}^{\infty} dx \exp(-x^2) = \sqrt{\pi}. \tag{1}$$

Всё остальное Вы узнаете,
прочитав эту книгу.

1

Рис. 1.1. Пример печатного документа

Программа `pdflatex` на выходе создаст файл `first.pdf`. Его можно просмотреть на экране или напечатать при помощи программы Adobe Reader, бесплатно распространяемой компанией Adobe, или при помощи другой широко известной программы GSview. Эти программы поставляются отдельно от системы LaTeX. При печати на бумагу PDF-документ имеет столь же высокое качество, как и DVI-документ, но его проще пересылать по электронной почте или экспонировать в интернете, поскольку программы Adobe Reader и GSview могут встраиваться в наиболее распространённые Web обозреватели.

Здесь уместно спросить, а стоило ли тогда вообще упоминать про файлы `dvi` и компилятор `latex`, если `pdf` и `pdflatex` имеют столь неоспоримое преимущество. Действительно, оснований для этого остаётся всё меньше и меньше. Последнее серьёзное препятствие для повсеместного перехода от `latex` к `pdflatex` исчезло в конце 2001 года, когда Владимир Волович распространил PostScript-версию русских шрифтов для LaTeX'а. Шрифты METAFONT, технология которых была разработана Д. Кнутом специально для TeX'а, малопригодны для использования с программой `pdflatex`, так как их можно внедрить в документы PDF только с потерей качества. Подробнее мы затронем эту тему в главе 16, а теперь перейдём к анализу входного файла в нашем примере.

В преамбуле `first.tex` имеются пять команд в строках 6–12, но лишь первые три из них необходимо располагать именно в преамбуле. Команды `\title` (заголовок), `\author` (автор) и `\date` (дата) могут находиться после `\begin{document}`, но до команды `\maketitle`, которая собственно и печатает заголовок, фамилию автора и дату.

Команды

```
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
\usepackage{indentfirst}
```

достойны более подробного обсуждения. Если `\documentclass` определяет основную структуру печатного документа, то команда `\usepackage` вносит в неё дополнения и изменения, порой очень значительные. Команда `\usepackage` загружает *пакет*. Пакет — это служебный текстовый файл с расширением `sty`. В данном случае загружаются файлы `inputenc.sty`, `babel.sty` и `indentfirst.sty`.

Пакет inputenc используется для объявления (ещё говорят: для декларирования) *кодировки* исходного текста. LaTeX способен компилировать входные файлы, подготовленные на самых разных компьютерных платформах, и, соответственно, имеющие самые разные кодировки. Кодировка — это порядок следования букв в «компьютерном алфавите». В этом «алфавите» гораздо больше букв, чем в обычном. В частности, строчные и прописные буквы имеют разные коды. Цифры и знаки препинания также считаются частью компьютерного алфавита. Прибавьте к этому буквы из других языков, ведь современные компьютеры — полиглоты. В нашем примере объявлена кодировка `cp1251`. Это означает, что текст во входном файле записан в кодировке, используемой русскими версиями операционной системы Windows. Для старых текстов, подготовленных в Microsoft DOS, следовало

бы указать кодировку cp866. Если текст получен из компьютера, работающего под управлением операционной системы Unix, то скорее всего нужно выбрать кодировку koi8-ru.

Главная задача пакета babel — включить алгоритм переноса русских слов. Он также заменяет русскими эквивалентами все английские слова, которые компилятор автоматически вставляет в печатный документ. Например, слово «Chapter» перед началом главы заменяется на слово «Глава». То, что пакет babel должен включить настройки именно русского языка, указывает опция russian в необязательном аргументе команды \usepackage. В многоязычных документах нужно перечислить все используемые языки. Например, в научных статьях на русском языке, как правило, ещё используют английский язык в качестве вспомогательного:

```
\usepackage[english,russian]{babel}
```

По ходу изложения мы обсудим множество проблем, связанных с русификацией LaTeX'а. С первой из таких проблем Читатель встретится в разделе 1.5.

Пакет indentfirst делает отступ в начале первого абзаца после заголовка документа или заголовка раздела, как того требует отечественная издательская культура. В документах на других языках отступ в начале первого абзаца часто не делают и пакет indentfirst не загружают.

Имеется множество стандартных пакетов, входящих в минимальный комплект поставки системы LaTeX, и ещё больше разного рода «самоделок». Стандартными являются inputenc и babel, а также пакеты graphics и graphicx (для импортирования графических изображений), color (для работы с цветом), longtable (для набора многостраничных таблиц) и многие другие. Краткая характеристика стандартных пакетов дана в разделе 3.3.

1.4. Буквы и символы

Большинство команд LaTeX'а описывают логические структуры. Входной файл содержит структуры совершенно различных размеров — от отдельной буквы до текста документа в целом. Начнём с букв.

LaTeX распознаёт строчные и прописные буквы, десять цифр от 0 до 9 и шестнадцать знаков препинания:

```
. , : ; ? " ‘ ’ ( ) [ ] - / * @
```

причём ‘ и ’ используются в качестве левой и правой кавычек. Символ " в некоторых случаях печатается в виде двойных правых кавычек, но не имеет левого аналога, поэтому вряд ли его стоит использовать, поскольку двойные кавычки можно набрать, удвоив одинарные.

Пять символов:

```
+ = | < >
```

применяются в таблицах и математических формулах, хотя + и = допускаются также и в обычном тексте.

Ещё десять символов:

```
\ # $ % & { } _ ^ ~
```

зарезервированы для служебного пользования. Как получить эти символы в печатном документе, рассказано в главе 4.

Помимо перечисленных имеются ещё невидимые символы, а именно: пробел и признак конца строки, которые вводятся в файл соответственно при нажатии клавиш ⟨Space⟩ (пробел) и ⟨Enter⟩ (ввод). Эти и некоторые другие невидимые символы (например, табулятор) интерпретируются LaTeX'ом совершенно одинаково. Поэтому мы также будем называть их *пробелами* и при необходимости изображать символом ␣. Число следующих подряд пробелов несущественно: сто пробелов интерпретируются как один, если только они не составляют *пустую* строку. Пустой является строка, в которой кроме признака конца строки могут быть только другие пробелы. Пустая строка метит конец абзаца.

Служебный символ \ (обратный слеш) имеет особое значение в LaTeX'е. С него начинаются *команды*. Большинство команд, начинаясь с обратного слеша, состоят из одной или более букв. Считывая входной файл, компилятор знает, что дошёл до конца такой команды, если встретил символ, не являющийся буквой: цифру, знак препинания, служебный символ, пробел или конец строки. Чаще всего такая команда заканчивается пробелом. Поэтому компилятор игнорирует *все* пробелы, следующие за ней.

Ну вот и всё, что можно рассказать про буквы и символы. Правда, мы не перечислили символы, которые являются ещё и буквами. «Не велика беда,— скажет Читатель.— Это и так ясно: a, b, c, ... x, y, z; а, б, в, ... э, ю, я». Скажет — и будет не прав. Дело в том, что LaTeX разделяет все символы на несколько *категорий*. Так вот: русским буквам присвоена категория так называемых активных символов. Проще говоря, русские буквы являются командами. Между прочим, это означает, что русские буквы нельзя использовать в именах других команд. Такая ситуация существовала не всегда, но мы условились не обсуждать альтернативные проекты, если они того не заслуживают[5]. Мы вернёмся к обсуждению этого вопроса в разделе 2.1. «Права» русских и латинских букв можно уравнять, используя механизм TCX, описанный в разделе 16.5.2, хотя ценой такого равенства будет потеря совместимости со стандартной версией системы LaTeX.

1.4.1. Знакомимся с клавиатурой

Все вышеперечисленные символы нетрудно отыскать на клавиатуре персональных компьютеров. Исторически существовало несколько способов размещения

[5] В недавно вышедшем третьем издании книги С. М. Львовского [16] описывается его собственная русификация, не совместимая со стандартной версией системы LaTeX, распространяемой через CTAN. Выбирая нестандартную версию, Читатель может столкнуться с непредвиденными проблемами, в том числе с невозможностью расширения набора используемых шрифтов.

русских букв на клавиатуре. В наши дни преобладает раскладка, максимально приближенная к клавиатуре печатной машинки. На старых клавиатурах не все буквы русского алфавита соответствуют их изображению на экране дисплея. Но пользователи таких компьютеров заведомо знают, как справиться с этой проблемой. Каждому символу на экране дисплея соответствует код, т. е. порядковый номер в кодовой таблице (странице) операционной системы, установленной на компьютере. В MS DOS использовалась так называемая альтернативная кодировка, которая по классификации корпорации Microsoft соответствовала кодовой странице 866. В русских версиях Windows принята кодировка ANSI 1251. В операционной системе Unix используется кодировка KOI8-RU или KOI8-R.

Шрифты, которые использует LaTeX для печати документа и его просмотра на экране дисплея, имеют собственную (внутреннюю) кодировку, не совпадающую в точности ни с одной из перечисленных. Это результат длительного эволюционного развития системы. К множественности кодировок входных файлов LaTeX приспособился путём развития собственных средств перекодирования исходных текстов. Такое перекодирование выполняет пакет inputenc, который описан в разделе 16.5.2.

Предполагается, что проблемы с кодовыми страницами исчезнут после повсеместного перехода на кодировку Unicode, в которой есть место для 65 000 символов. Некоторые текстовые редакторы, в том числе `notepad` (блокнот) в последних версиях Windows, позволяют сохранять тексты в кодировке Unicode, однако LaTeX не принимает входные файлы Unicode.

1.4.2. Шрифты, литеры и лигатуры

LaTeX автоматически делает *кернинг* сочетаний букв и распознаёт *лигатуры*.

Кернение — это способ печати, который регулирует средний промежуток между соседними *литерами*[6], т. е. изображениями букв (символов). Этот промежуток зависит от ширины литер. Некоторые пары литер смотрятся лучше, когда они стоят более тесно, другие — когда раздвинуты.

Лигатура — это комбинация символов, изображаемая одной литерой. Например, комбинации `ff`, `fi`, ``` `` ```, `''`, `<<` и `>>` печатаются лигатурами ﬀ, ﬁ, “, ”, « и ». Информация, необходимая для кернинга и выделения лигатур, заложена в файле метрики шрифтов.

Первое семейство шрифтов для LaTeX'а создано Дональдом Кнутом и называется Computer Modern, или просто CM. На смену CM разработчики LaTeX 2_ε предложили шрифты, обозначаемые при помощи аббревиатуры EC. Внешне они похожи на шрифты CM и отличаются от них только внутренней кодировкой. Внутренняя кодировка шрифтов, используемых в издательской системе LaTeX, не имеет ничего общего с кодировкой входного файла. Благодаря этому LaTeX способен компилировать исходные тексты с произвольной кодировкой. Кодировку шрифтов CM сейчас обозначают как `OT1`, а кодировку шрифтов EC как `T1`.

[6] Здесь и далее мы используем термин *литера* в значении, несколько отличном от принятого в типографском деле. В англоязычной литературе употребляется слово *glyph*.

Современные шрифты для LaTeX'а содержат до 256 литер, тогда как шрифты CM содержали не более 128. В первых русских версиях LaTeX'а к 128 символам Computer Modern добавляли ещё русские буквы, так что общее число литер в шрифте доходило до тех же 256. В кодировке `T1` для русских букв нет места, поэтому для них специально введена кодировка `T2A`. Пакет babel с опцией `russian` загружает шрифты именно с этой кодировкой. Кириллические шрифты разработаны Ольгой Лапко и обозначаются аббревиатурой LH.

Шрифты CM, EC и LH объединяет общность происхождения. Все они генерируются системой METAFONT, разработанной Д. Кнутом [3]. Поэтому их иногда называют `mf`-шрифтами, по расширению имени файла, где записана программа генерации шрифта.

Уже после изобретения системы METAFONT получили распространение шрифты PostScript, TrueType и OpenType. Их также можно приспособить для печатных документов LaTeX, загрузив один-два дополнительных пакета. Мы расскажем о таких пакетах в главе 16.

Владимир Волович перевел шрифты EC и LH в форму PostScript, присвоив им имя CM-Super. Шрифты CM-Super позволяют практически любой исходный текст с разметкой LaTeX откомпилировать в документ PDF без потери качества, которая случается при внедрении в документ PDF шрифтов METAFONT. При этом не нужно загружать дополнительные пакеты, если, конечно, нет желания использовать шрифты с совершенно иными визуальным свойствами, нежели Computer Modern.

Знатоки утверждают, что шрифты, происходящие от Computer Modern, не очень красивы. Читатель сам может вынести своё суждение на этот счёт, — весь текст нашей книги, кроме главы 16, набран шрифтами CM-Super.

1.4.3. Специальные символы

Широкий набор символов, описанный в главе 4, позволяет легко создавать тексты нестандартного написания. Фраза на французском языке «Nous sommes prêts à partir pour l'Université» во входном файле может выглядеть примерно так:

```
Nous sommes pr\^{e}ts \`{a} partir pour l'Universit\'{e}
```

1.5. Слова и предложения

LaTeX игнорирует то, как набран исходный текст во входном файле. Он фиксирует только границы слов, предложений и абзацев.

Слова, как обычно, отделяются друг от друга одним или несколькими пробелами.

Предложение заканчивается точкой (`.`), восклицательным (`!`) или вопросительным (`?`) знаками, за которыми следует пробел. После этих знаков LaTeX обычно чуть увеличивает пробел. Однако отечественные типографские правила

не предусматривают подобного увеличения, поэтому пакет babel по умолчанию отменяет такое увеличение для русского языка (см. раздел 4.3).

Пустая строка означает конец абзаца.

Форматирование текста не зависит от числа пробелов или пустых строк: лишние просто игнорируются. Примером может служить входной файл из раздела 1.3.

1.5.1. Кавычки

В машинописном тексте правые и левые кавычки печатаются совершенно одинаково: ". LaTeX изображает символ " как прямые (") или правые кавычки (”) в зависимости от используемого шрифта. Чтобы напечатать левые кавычки (“), набирают два символа ` подряд: ``. Тогда уж для единообразия правые кавычки (“) набирают в виде двух апострофов: ''. В текстах на английском и других иностранных языках встречаются также одинарные кавычки ` и ':

```
Знаки препинания в конце цитаты
принято выносить за ``кавычки'',
чтобы не получилось `некрасиво.'
```

Знаки препинания в конце цитаты принято выносить за “кавычки”, чтобы не получилось ‘некрасиво.’

В редких случаях, когда цитата сама содержит цитату и нужно «раздвинуть» кавычки, это можно сделать, добавив небольшой пробел с помощью команды `\,`:

```
``\,`Вадан', --- говорит наш сын.
Попробуй догадайся, что это
`диван'\,''.
```

“ ‘Вадан’, — говорит наш сын. Попробуй догадайся, что это ‘диван’ ”.

Ядро LaTeX'а не содержит команд, соответствующих русским кавычкам в виде «ёлочек», но пакет babel с опцией `russian` обеспечивает один-два подходящих способа. Удобнее всего набирать такие кавычки при помощи лигатур << и >>:

```
<<Дети --- цветы жизни>>
```

«Дети — цветы жизни»

Кавычки в виде „лапок“ также удобно набирать при помощи лигатур ,, и ``:

```
,,Упрямство --- вывеска дураков``.
```

„Упрямство — вывеска дураков“.

Если в начале или в конце цитаты или прямой речи встречаются внутренние и внешние кавычки, то они должны различаться между собой.

```
ТАСС сообщает: <<,,Баллада о солдате``
получила 2 премии международного
кинофестиваля в Канне>>.
```

ТАСС сообщает: «„Баллада о солдате“ получила 2 премии международного кинофестиваля в Канне».

Иные виды кавычек перечислены в разделе 4.2.

1.5.2. Дефисы и тире

Можно напечатать тире трёх различных размеров, если набрать один, два или три символа «-» подряд. LaTeX печатает «--» и «---» как лигатуры. Одинарный дефис используют в составных словах:

```
как-то, где-нибудь.
```

как-то, где-нибудь.

Двойной дефис рекомендуется для указания диапазона чисел:

```
3--4, 1903--1991.
```

3–4, 1903–1991.

Тройной дефис означает тире:

```
Это --- слон.
```

Это — слон.

В текстах на английском языке пробелы между тире и окружающими словами удаляют:

```
``Are you---are you
fond---of---of dogs?''
```

“Are you—are you fond—of—of dogs?”

«Иностранное» тире имеет бо́льшую длину — соответствующие настройки автоматически производятся при переключении языка (раздел 3.6).

Знак «минус» не является дефисом, хотя во входном файле они изображаются одним и тем же символом. «Минус» может появляться только в математических формулах, которые мы обсудим в главе 6. LaTeX использует особые правила кернинга математических формул.

1.5.3. Логосы

Отдельные слова и даже предложения могут набираться специальными командами — логосами. Логосы TeX, LaTeX и LaTeX 2_ε печатаются командами `\TeX`, `\LaTeX` и `\LaTeXe` соответственно.

Другой полезной командой является `\ldots`. Она печатает многоточие, то есть последовательность трёх точек, заменяющую пропущенный текст. Простой набор трёх точек подряд не создаст многоточие с правильным расстоянием между точками:

```
Сравните ... с \ldots
```

Сравните ... с . . .

Команда `\today` генерирует сегодняшнюю дату:

```
Эта страница напечатана
\today Запомним это!
```

Эта страница напечатана 30 апреля 2004 г.Запомним это!

Напомним, что LaTeX игнорирует все пробелы, следующие за командой, состоящей из букв. Именно по этой причине перед словом «Запомним» в последнем примере исчез пробел. Если всё-таки пробел необходим, нужно вставить команду `\␣` (где ␣ обозначает пробел):

```
Эта страница напечатана
\today\ Запомним это!
```

Эта страница напечатана 30 апреля 2004 г. Запомним это!

LaTeX различает прописные и строчные буквы в имени команд. Написав `\Today`, Читатель совершит ошибку. Большинство команд содержит только строчные буквы.

1.5.4. Подстрочные примечания

Подстрочное примечание (сноску) печатает команда `\footnote`. Её нужно вставить в том месте исходного текста, где в печатном документе должен появиться маркёр примечания, а в качестве аргумента набрать текст примечания. Подстрочное примечание[7] на этой странице во входном файле набрано следующим образом:

```
Подстрочное примечание\footnote{Мы иногда пренебрегаем
правилами пунктуации. Просим извинить нас.} на этой...
```

Маркёр сноски печатается там, где стоит обратный слеш \, начинающий команду `\footnote`. Поэтому между словом `примечание` и командой `\footnote` в данном примере нет пробела — иначе пробел появился бы перед маркёром.

Способ маркировки подстрочного примечания определяется классом документа. Обычно сноски нумеруются арабскими цифрами и печатаются внизу страницы, а не в конце документа. Класс `article` определяет, что используется сквозная нумерация сносок на всём протяжении документа. В классах `report` и `book` ссылки нумеруются независимо в пределах каждой главы. При удалении или добавлении сноски все другие перенумеровываются автоматически.

Команда `\footnote` не может использоваться в аргументах большинства других команд. В главе 4 объяснено, как создать подстрочное примечание к тексту, который содержится в аргументах каких-либо команд.

1.6. Комментарии

Как и любой другой язык программирования, LaTeX имеет средства для комментирования содержимого входного файла. LaTeX игнорирует любой текст между символом `%` и концом текущей строки. Трудно представить ситуацию, когда нужно было бы комментировать окончательный текст печатного документа, однако в процессе работы над ним такая необходимость возникает довольно часто. Любителям хранения «мусора» во входном файле комментарии мало помогут, но им полезно знать, что LaTeX воспринимает как комментарий всё, что во входном файле находится после команды `\end{document}`; туда можно временно складывать ненужные фрагменты текста.

Помимо комментирования, символ `%` может иметь более важное применение: часто он необходим, чтобы начать новую строку во входном файле без ввода пробелов в конце предыдущей. Именно такой эффект демонстрирует следующий пример, с небольшим изменением позаимствованный со страницы 26:

```
‘‘Are you---are you fond---%
of---of dogs?’’
```

“Are you—are you fond—of—of dogs?”

[7] Мы иногда пренебрегаем правилами пунктуации. Просим извинить нас.

1.7. Строки и абзацы

Пустая строка, не содержащая даже комментария, означает конец абзаца. Количество пустых строк не имеет значения. Так же не имеет значения, начинается первая строка в абзаце с первой или тринадцатой позиции — LaTeX вставит в печатный документ отступ фиксированной длины. С отступа начинается первая строка каждого абзаца, однако стандартные классы определяют, что первый абзац после заголовка отступа не имеет. Команда `\noindent` отменяет отступ, а `\indent`, наоборот, вставляет, но даже она бессильна перед магией первого абзаца — столь сильна традиция в английском языке не делать отступ после заголовка. Мы уже упоминали о пакете indentfirst и в разделе 4.5 расскажем ещё об одном способе борьбы с этой традицией.

LaTeX автоматически разбивает текст на строки и переносит слова так, чтобы пробелы между словами в соседних строках также были по возможности равны. Однако он не способен определить логически связанные сочетания слов, которые желательно располагать на одной строке. Например, выражение «рис. 3» выглядело бы странно, если бы «рис.» заканчивал одну строку, а «3» начинал следующую. Символ «~», вставленный между соседними словами, создаёт пробел, на котором LaTeX никогда не разорвёт строку:

```
На рис.~3 гр.~Вострикову и Ф.И.~Глазову от~2 до~5 лет.
```

В текстах на русском языке новая строка не должна начинаться с тире. Чтобы гарантировать это, пробел перед тире делают неразрывным:

```
Это он, это он~--- ленинградский почтальон.
```

Это он, это он — ленинградский почтальон.

Столь же просто установить запрет на перенос отдельного слова по слогам или обучить LaTeX правильно переносить слова, которых он не знает. Желающих освоить эту простую науку мы отсылаем к разделу 4.4.

1.8. Выделение текста

Вслед за разделами «Буквы и символы», «Слова и предложения», «Строки и абзацы», по логике вещей, должны следовать «Страницы и . . . ». Но это был бы слишком большой шаг вперёд. Чтобы не делать затем два шага назад, расскажем сначала, какими средствами располагает LaTeX для выделения фрагментов печатного текста.

В машинописном тексте слова, несущие особую смысловую нагрузку, выделяют подчеркиванием. LaTeX умеет подчеркивать. Однако в типографском документе для выделения текста обычно используют *курсив*. Курсивный текст печатает команда `\emph`. Чтобы выделить курсивом какой-либо фрагмент текста, его нужно сделать аргументом этой команды:

```
...обычно используют \emph{курсив}. Курсивный...
```

В выражении `\emph{курсив}` обратный слеш и следующие за ним четыре буквы `\emph` составляют *имя команды*, а `{курсив}` — её *аргумент*. Большинство команд либо не имеют аргументов, как `\today`, либо имеют один аргумент, как `\emph{...}`. Есть, впрочем, некоторое количество более сложных команд, имеющих несколько аргументов, причём каждый заключён в фигурные скобки, а пробелы между аргументами и между именем команды и первым аргументом игнорируются.

Команды типа `\emph` могут быть вложены друг в друга вполне очевидным способом. В стандартных классах документов для выделения текста внутри уже выделенного текста используется прямой шрифт:

```
Выделенный \emph{текст
\emph{внутри} курсива}
печатается прямым шрифтом.
```

Выделенный *текст* внутри *курсива* печатается прямым шрифтом.

Точнее говоря, команда `\emph` переключает шрифт на курсивный, если до неё использовался прямой шрифт. В остальных случаях она включает прямой шрифт. Не следует злоупотреблять чрезмерно частым выделением фрагментов текста. Это рассеивает внимание читателя, вызывая эффект, обратный желаемому. Есть много других вариантов переключения шрифта. Мы обсудим их после того, как объясним, что такое *декларация*.

1.9. Декларации

Для выделения текста вместо `\emph{...}` можно использовать конструкцию `{\em ...}`:

```
Выделенный {\em текст {\em внутри}
курсива} печатается прямым шрифтом.
```

Выделенный *текст* внутри *курсива* печатается прямым шрифтом.

Команда `\em` не имеет аргументов, а просто информирует LaTeX, что надо начать выделять текст. Она не производит ни текст, ни пробел. Такие команды называются *декларациями*. Большинство команд, определяющих внешний вид печатного документа (т. е. его *класс*), также являются декларациями, в том числе команды `\documentclass` и `\usepackage`. Декларация преобразует способ действия других команд или же создаёт другие команды. В частности, декларация `\em` инструктирует LaTeX, что нужно сменить шрифт так, чтобы он отличался от используемого в настоящий момент. Область действия декларации, как показывает приведённый пример, можно выделить, группируя текст в *блок* с помощью фигурных скобок. Левая скобка `{` открывает новый блок, а правая скобка `}` закрывает текущий блок. Для каждой открывающей скобки `{` во входном файле должна иметься парная ей закрывающая скобка `}`. В следующем примере, из которого удалён весь текст, кроме фигурных скобок, парные скобки помечены одинаковыми индексами: $\{_1 \quad \{_2 \quad \}_2 \quad \{_3 \quad \{_4 \quad \}_4 \quad \}_3 \quad \}_1$.

Когда скобка { открывает новый блок, все декларации, действовавшие до неё, остаются в силе до тех пор, пока не встретится противоречащая им декларация. Закрывающая фигурная скобка } отменяет действие всех деклараций, стоящих после парной ей открывающей фигурной скобки. В следующем разделе мы увидим, что область действия декларации ограничивают также командные скобки \begin и \end.

1.10. Процедуры

Если нужно выделить большой фрагмент текста, то использование команды \emph или декларации \em может вызвать непредвиденные трудности, так как входной файл станет трудночитаемым. Иногда совсем не просто найти закрывающую фигурную скобку }, если она очень далеко отстоит от парной ей открывающей скобки {.

Многие декларации имеют аналоги в виде процедур с теми же именами, но без обратного слеша в имени. Так, декларации \em соответствует процедура em. Любая процедура начинается с *командной скобки* \begin{*env*} и закрывается командной скобкой \end{*env*}, где *env* — имя процедуры, в данном случае — em.

```
...обычно используют
\begin{em} курсив. \end{em}
Курсивный...
```

...обычно используют *курсив.* Курсивный...

Текст между командными скобками составляет *тело процедуры* и форматируется в соответствии с тем, как она определена. Процедуры могут центрировать текст, выравнивать его по правой или левой границе, печатать в виде списков, таблиц, рисунков, формул и т. д. Командные скобки \begin{document} и \end{document} выделяют текст самого документа. Всё, что находится после команды \end{document}, попросту игнорируется, а команды, предшествующие \begin{document}, обычно определяют класс документа.

Командные скобки \begin и \end ограничивают область действия деклараций наравне с фигурными, как показывает следующий пример с процедурой center, о которой пойдет речь ниже:

```
\begin{center}
  Этот текст автоматически
  \em центрируется.
\end{center}
А этот~--- нет.
```

Этот текст автоматически
центрируется.

А этот — нет.

Декларации полезно оформлять в виде процедуры, если они действуют на большие фрагменты текста.

LaTeX имеет множество процедур и допускает создание новых с помощью команды \newenvironment, описанной в главе 7. Во вводной главе мы ограничимся перечислением основных процедур, чтобы помочь Читателю выбрать нужные и найти главу, в которой они описаны.

- Процедура `center` центрирует текст, а `flushleft` и `flushright` выравнивают текст соответственно по левой и правой границам страницы (глава 5).

- Процедуры `quote` и `quotation` используются для *цитирования* текста. Можно сказать, что они выполняют роль кавычек для длинных цитат. Обычно цитированный текст печатается с небольшим отступом от правой и левой границ страницы (глава 5).

- Процедура `verbatim` печатает текст машинописным шрифтом так, как он сформатирован во входном файле. Текст учебного файла в разделе 1.3 напечатан процедурой `verbatim` (глава 5).

- Процедура `verse` применяется для печати стихов (глава 5).

- Процедуры `itemize`, `description` и `enumerate` применяются для составления списков. Данный список *записей*, помеченных знаком •, составлен процедурой `itemize` (глава 5).

- Процедуры `math`, `displaymath`, `equation` и `eqnarray` печатают математические формулы, причём последние две ещё и автоматически нумеруют их (глава 6).

- В процедуре `picture` доступны особые команды рисования (глава 9).

- Процедура `minipage` формирует *боксы*. Основное свойство бокса состоит в том, что его нельзя разрезать и нельзя по частям перенести со страницы на страницу. Все примеры в этой книге, содержащие вертикальную черту посредине, состоят из двух боксов — министраниц, сформированных процедурой `minipage` (глава 9).

- Процедуры `table` и `figure` создают *плавающие объекты*. Они используются для размещения таблиц и рисунков, а также для создания подписей к ним (глава 11).

- Процедуры `tabbing` и `tabular` применяются для составления таблиц. Все таблицы в этой главе составлены с помощью процедуры `tabular` (глава 12).

- Процедура `thebibliography` печатает список литературы, позволяя возложить на LaTeX нумерацию ссылок (глава 13).

- Процедура `theindex` формирует алфавитный указатель (глава 14).

- Пометив с помощью команды `\label{`*key*`}` какой-нибудь меткой *key* любой пронумерованный объект (например, формулу, таблицу и т. д.), можно напечатать номер этого объекта в любом месте текста при помощи команды `\ref{`*key*`}` (глава 3).

Дополнительные возможности открывает использование пакетов, загружаемых командой `\usepackage` в преамбуле входного файла. Пакеты образуют важное дополнение к процедурам и командам, составляющим ядро (формат) LaTeX'а, которое автоматически загружается перед компиляцией входного файла. Они обеспечивают подготовку документов на разных языках, позволяют импортировать рисунки, созданные специализированными графическими программами, помогают форматировать сложные или цветные таблицы, подключать новые шрифты и т. д. С разной степенью полноты мы опишем более 100 пакетов. Их краткие аннотации и указания на соответствующие главы книги приведены в разделе 3.3.

1.11. Экскурсия в море шрифтов

Выбор характеристик шрифта подчинён логической структуре текста. В нашей книге выделенный текст набран *курсивом*, а текст, предназначенный для буквального воспроизведения во входном файле, набран `машинописным шрифтом`. LaTeX различает форму шрифта (shape), серию (series) и семейство (family). На языке профессионалов форма шрифта называется начертанием, серия — насыщенностью, а семейство — гарнитурой (глава 16).

Начертание	
Upright shape (прямое)	`\textup{Upright shape ...}`
Italic shape (курсивное)	`\textit{Italic ...}`
Slanted shape (наклонное)	`\textsl{Slanted ...}`
SMALL CAPS SHAPE (КАПИТЕЛЬ)	`\textsc{Small caps ...}`
Насыщенность	
Medium series (средняя)	`\textmd{Medium series ...}`
Boldface series (полужирная)	`\textbf{Boldface ...}`
Гарнитура	
Roman family (романская)	`\textrm{Roman family ...}`
Sans serif family (рубленая)	`\textsf{Sans serif ...}`
`Typewriter family (машинописная)`	`\texttt{Typewriter ...}`

По умолчанию используется шрифт формы upright, серии medium, семейства roman, поэтому в примере выше три команды `\textup`, `\textmd` и `\textrm` выбрали один и тот же шрифт upright-medium-roman. Однако, комбинируя команды, можно получить большой набор вариантов:

```
\textsf{Кто, \textbf{Где}}
\textit{и \texttt{Когда}}
вздумает использовать полужирный
рубленый шрифт?
```

Кто, **Где** *и Когда* вздумает использовать полужирный рубленый шрифт?

Некоторые комбинации могут быть недоступны на компьютере Читателя. Если заказана недоступная комбинация, компилятор выведет на экран предупрежда-

ющее сообщение и заменит её на ту, которая, по его разумению, ближе всего соответствует заказанной.

Каждой из перечисленных выше команд соответствует декларация. Например, полужирный текст можно набрать либо при помощи команды `\textbf`, либо при помощи декларации `\bfseries`:

```
Всё \textbf{выше}, и {\bfseries
выше}, и {\bfseries выше}!
```

Всё **выше**, и **выше**, и **выше**!

Соответствие команд и деклараций показывает табл. 1.1.

Ни одна из этих команд или деклараций не может использоваться в математических формулах, где для изменения шрифта применяются другие команды. Они будут перечислены в разделе 6.6. Набор шрифтов, применяемых в математических формулах, несколько отличается от шрифтов для обычного текста. Например, среди математических шрифтов обычно нет капительного начертания, зато имеются каллиграфические символы.

Выбор шрифта есть признак визуального проектирования печатного документа. Команды, детализирующие визуальные свойства печатного документа, вообще-то не предназначены для «имплантации» в текст. Их следует использовать в определениях команд, описывающих логическую структуру текста, например в командах, печатающих заголовок документа или его раздела. Отчасти по этой причине команды переключения шрифтов имеют столь длинные и неудобные имена. LaTeX предлагает команду `\emph` для логического выделения небольших фрагментов текста, и этого вполне достаточно. В разделе 7.1 мы расскажем, как определить свои собственные команды для выделения логических структур в тексте. Для этой цели служит декларация `\newcommand`. Допустим, автор желает, чтобы названия животных в его книге печатались курсивом. Тогда он может определить команду `\animal`, которая будет делать то же самое, что и команда `\textit`:

```
\newcommand{\animal}{\textit}
\animal{Слон} и \animal{Муравей}
```

Слон и *Муравей*

Если через некоторое время автор решит, что слова *Слон* и *Муравей* лучше напечатать машинописным шрифтом, он сможет легко изменить определение команды `\animal`. Не следует писать `\textit` вместо `\animal`, так как команда `\textit` может понадобиться для выделения терминов иного сорта. Хотя команда `\animal` не короче, чем `\textit`, зато она ясно напоминает, для чего предназначена, так как в переводе на русский слово animal означает животное.

Имеется также набор деклараций для изменения размера, т. е. *кегля* шрифта. Они приведены в табл. 1.2. По умолчанию (если кегль не указан явно) действует декларация `\normalsize`. Любая декларация, изменяющая размер шрифта, не влияет ни на его начертание, ни на насыщенность, ни на гарнитуру. И наоборот, при изменении вида шрифта его размер не меняется. Отметим, что в предыдущей версии LaTeX'а ситуация была прямо противоположной. Вместо команд и деклараций, перечисленных в табл. 1.1, имелось всего 7 деклараций, перечисленных в

Таблица 1.1

Команды и декларации переключения шрифтов

Команда	Декларация	Команда	Декларация	Команда	Декларация
\textup	\upshape	\textmd	\mdseries	\textrm	\rmfamily
\textit	\itshape	\textbf	\bfseries	\textsf	\sffamily
\textsl	\slshape			\texttt	\ttfamily
\textsc	\scshape				

Таблица 1.2

Декларации переключения размера (кегля) шрифта

Декларация	Название	Образец
\tiny	крошечный	Аа. . . яЯ
\scriptsize	индексный	Аа. . . яЯ
\footnotesize	подстрочный	Аа. . . яЯ
\small	маленький	Аа. . . яЯ
\normalsize	стандартный	Аа. . . яЯ
\large	большой	Аа. . . яЯ
\Large	Большой	Аа. . . яЯ
\LARGE	БОЛЬШОЙ	Аа. . . яЯ
\huge	огромный	Аа. . . яЯ
\Huge	Огромный	Аа. . . яЯ

Таблица 1.3

Устаревшие декларации переключения шрифтов,
поддерживаемые для совместимости с версией LaTeX 2.09.

Декларация	Название	Образец
\rm	Romanic	Романский
\it	*Italic*	*Курсив*
\bf	**Bold face**	**Полужирный**
\sc	Small caps	Капитель
\sf	Sans serif	Рубленый
\sl	*Slanted*	*Наклонный*
\tt	Typewriter typefaces	Машинописный

табл. 1.3, взаимно исключающих друг друга. Это означает, что, к примеру, декларация `\bf` отменяет действие декларации `\it`, тогда как результат действия деклараций `\bfseries` и `\itshape` «перемножается»:

```
Сравните {\bf A\it Б} и
{\bfseries A\itshape Б}.
```

Сравните **A***Б* и **A*Б***.

В новой версии LaTeX'а — LaTeX 2_ε — декларации из табл. 1.3 не определены в *формате* LaTeX'а, но стандартные классы поддерживают их для совместимости с прежней версией LaTeX 2.09.

Действие деклараций переключения размера шрифта в версии LaTeX 2.09 также отличалось от принятой ныне. Эти декларации помимо изменения размера шрифта одновременно устанавливали романский прямой шрифт `\rm`. Теперь эти нелогичные странности остались в прошлом. Однако приверженцы старины могут заставить LaTeX эмулировать правила, действовавшие в версии 2.09, с помощью пакета newlfont или oldlfont (приложение A).

Размер шрифта зависит также от класса документа, который определяет конкретное значение каждой декларации из табл. 1.2. Опции `11pt` и `12pt` в команде `\documentclass` увеличивают размер используемых шрифтов в среднем на 10% и 20% соответственно по сравнению с опцией `10pt`, используемой по умолчанию. Название опций `10pt`, `11pt`, `12pt` отражает размер (кегль) основного шрифта печатного документа, выраженный в пунктах (единицах типографской системы мер, равных 0,376 мм). Например, данная книга набрана десятым кеглем. Разумеется, заголовки разделов оформлены шрифтом большего размера, а подстрочные примечания — меньшего, но основной текст, которому соответствует декларация `\normalsize`, имеет размер `10pt`. Подробнее соответствие между декларациями и размером шрифта обсуждается в главе 16 (раздел 16.3.5). Она предназначена для опытных пользователей. В частности там объясняется, как имена аргументов команд, например *class* или *options*, мы печатаем прямым курсивом. Понятно, что для этой цели не годится комбинация команд `\textup` и `\textit`, так как обе они меняют одну и ту же характеристику шрифта — его начертание, поэтому одна команда отменяет действие другой:

```
\textup{Class и \textit{Options}}
```

Class и *Options*

1.12. Печатный документ

Теперь настало время перейти к обсуждению структуры наибольшего размера — всего печатного документа в целом. Поскольку весь текст документа следует за `\begin{document}`, предшествующая часть входного файла, т. е. *преамбула*, может содержать только декларации, а также определения команд и процедур, которые предполагается использовать для форматирования печатного документа. Бо́льшую часть необходимых деклараций и определений загружает команда `\documentclass`. Она устанавливает *класс* печатного документа. В примере из

раздела 1.3 выбран класс article (статья): \documentclass{article}. При этом компилятор загружает служебный файл article.cls. Файлы классов имеют расширение cls.

Полезно усвоить твёрдое правило — всегда начинать входной файл именно с команды \documentclass. Достойно упоминания лишь одно исключение из этого правила, а именно: команде \documentclass может предшествовать комментарий, каждая строка которого начинается с %.

Выбирая класс печатного документа, обычно указывают одну или несколько опций. Стандартные классы имеют десятки опций. Мы уже говорили об опциях 11pt и 12pt, которые увеличивают используемый шрифт в среднем на 10 и 20 процентов. Так же упоминавшаяся опция twocolumn форматирует печатный документ в две колонки. Опция twoside форматирует текст так, чтобы его было удобно печатать на двух сторонах листа, а затем переплетать. Есть набор опций для выбора формата бумаги, на которой будет напечатан документ. Размеры страницы печатного документа — так называемая *полоса набора* — зависят от этого формата. Стандартные классы по умолчанию предполагают, что документ будет печататься на бумаге размером 8,5 × 11 дюймов (216 × 279 мм). Это несколько отличается от принятого в России и Европе формата A4, равного 210 × 297 мм. Поэтому стандартные классы практически всегда следует включать с опцией a4paper:

```
\documentclass[a4paper]{article}
```

В разделе 3.2 перечислены все опции стандартных классов.

Наряду с \documentclass преамбула может содержать другие декларации и определения, которые по отдельности, как правило, вносят меньшие изменения в класс документа, чем любая из стандартных опций. Если же таких деклараций много и они часто используются в разных документах, их можно записать в отдельный файл с расширением sty. Такой файл называется *пакетом* и загружается декларацией \usepackage. Например, команда

```
\usepackage{graphics}
```

загружает пакет graphics. Он используется для вставки (импортирования) графических изображений (рисунков) в печатный документ. Пакеты могут иметь опции так же, как и классы.

1.12.1. Титульная страница

Печатный документ обычно начинается с титульной страницы, на которой располагаются его название, список авторов и, возможно, дата. В краткой статье (документе класса article) вся эта информация будет напечатана в начале первой страницы текста, а не на отдельном листе, как в монографии (документе класса book). В любом случае титульную страницу производит команда \maketitle. Она не имеет аргументов. Вся информация для печати титульной страницы должна

быть введена заранее. Название, фамилии авторов и дата указываются соответственно в аргументах команд `\title`, `\author` и `\date`. Фамилии авторов разделяются командой `\and`, но можно также писать их через запятую или пробел. Команды `\title`, `\author` и `\date` можно поместить в преамбулу, как это сделано в примере из раздела 1.3. Команда `\date` необязательна. Если она пропущена, LaTeX печатает текущую дату[8]. Однако команды `\title`, `\author` обязательны, если `\maketitle` действительно используется. Чтобы команда `\maketitle` не печатала дату или, например, фамилию автора, аргумент соответствующей подготовительной команды должен быть пустым: `\date{}`. Команды, печатающие на титульной странице дополнительную информацию (адрес автора и т. п.), описаны в главе 3.

```
\subsection{Глава, секция и др.}

\LaTeX\ автоматически генерирует
номер раздела. Пустая строка до
после команды секционирования не
имеет значения.
```

1.12.2. Глава, секция и др.

LaTeX автоматически генерирует номер раздела. Пустая строка до или после команды секционирования не имеет значения.

Печатный текст значительного размера разбивают на логически завершённые части — *разделы*: главы, секции, параграфы и т. д. Разделы образуют иерархическую структуру, каждый элемент которой начинается с одной из команд секционирования, которые перечислены ниже в порядке возрастания уровня вложенности (по строкам слева направо):

```
\part        \chapter
\section     \subsection     \subsubsection
\paragraph   \subparagraph
```

Класс документа определяет, какие из этих команд доступны. Например, в классе `article` старшей из доступных команд секционирования является `\section` (секция), а в классе `book` — команда `\chapter` (глава). В результате документ класса `article` легко включить в книгу в виде отдельной главы, то есть как раздел `\chapter`. Команда `\part` является необязательной в том смысле, что она не влияет на нумерацию разделов более низкого уровня.

Текст приложения (если таковое имеется) начинается с декларации `\appendix` и может содержать те же команды секционирования, что и основная часть текста. Декларация `\appendix` ничего не печатает, а только меняет способ нумерации разделов сообразно тому, как это принято для приложений.

Чтобы напечатать оглавление, в текст входного файла нужно вставить команду `\tableofcontents`. Когда LaTeX переносит название раздела из аргумента команды секционирования в оглавление, он проделывает с ним некие манипуляции, которые небезопасны для ряда команд LaTeX'а. Такие команды называются *хрупкими* в отличие от *устойчивых* команд, которым ничего не страшно.

---

[8] Все подобные утверждения справедливы при использовании одного из стандартных классов. Мы не будем более напоминать об этом.

Аргументы, которые куда-либо передаются, называются *подвижными*. Может случиться, что команда, помещённая в аргумент другой команды, отказывается работать. Весьма вероятно, что она хрупкая и находится в подвижном аргументе. Тогда её нужно защитить, поставив перед ней команду `\protect`. Хрупкие команды обсуждаются в разделе 2.7.

1.13. Диагностические сообщения

Не каждому новичку удаётся с первой попытки получить идеально оформленный и красиво напечатанный текст из-за неизбежных поначалу ошибок.

Если Читателю, вопреки нашим предсказаниям, удалось безошибочно ввести образец исходного текста во входной файл `first.tex`, предлагаем ему поэкспериментировать, удалив букву `a` из команды `\begin{equation}`. Эта команда находится в строке 50. Пометьте также эту строку своим комментарием:

```
\begin{eqution} % пропущена буква ‘a’ в слове equation
```

Назовём испорченный таким образом входной файл `second.tex`. Читатель уже знает, как выполнить компиляцию входного файла. На нашем компьютере мы запускаем компилятор `latex` через меню редактора. На компьютере нашего Читателя способ запуска компилятора, а также сообщения, которые он выводит на экран, могут немного отличаться от приводимых ниже. Напомним, что мы используем комплект программ MiKTeX, работающий под управлением операционной системы Windows.

Приступив к работе, `latex` выводит на экран дисплея сообщение[9]:

```
This is e-TeX, Version 3.141592-2.1 (MiKTeX 2.4)
entering extended mode
(second.tex
LaTeX2e <2001/06/01>
Babel <v3.7m> and hyphenation patterns for english, russian, dumylang, nohyphen
ation, loaded.
(C:\texmf\tex\latex\base\article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX document class
(C:\texmf\tex\latex\base\size10.clo)) (C:\texmf\tex\latex\base\inputenc.sty
(C:\texmf\tex\latex\cyrillic\cp1251.def)) (C:\texmf\tex\generic\Babel\babel.sty
(C:\texmf\tex\generic\Babel\russianb.ldf (C:\texmf\tex\generic\Babel\babel.def)
 (C:\texmf\tex\latex\cyrillic\t2aenc.def)))
No file second.aux.
```

Из него следует, что работает программа-компилятор из коллекции MiKTEX. Она загружает файл `second.tex` (об этом говорит третья строка сообщения). Версия формата LaTeX 2_ε указана в четвёртой строке. Она датирована 1 июня 2001 года. В пятой строке сообщается, что загружены таблицы переносов для английского

[9] Мы сохранили разбиение на строки, как на экране.

и русского языков. Какие именно таблицы переносов будут загружены, определяется при установке системы LaTeX на компьютер (см. приложение B.1). Далее следует перечень загружаемых служебных файлов, который начинается с файла класса `article.cls`.

Сообщение «`No file second.aux`» в последней строке предупреждает, что не найден один из служебных файлов. Такого рода предупреждения появляются при первой компиляции любого входного файла. Обрабатывая входной файл, `latex` одновременно записывает служебную информацию во вспомогательный файл с тем же именем `second`, что и входной файл, но присваивает ему расширение `aux`. При повторной обработке такой вспомогательный файл уже будет найден, и компилятор сообщит:

```
(second.aux)
```

Файлы с расширением `aux` используются для автоматического создания перекрёстных ссылок на номера страниц, рисунков, таблиц, цитируемую литературу и так далее. Так как в данном случае перекрёстных ссылок нет, файл `second.aux` в общем-то бесполезен.

Затем обычно на экран выводятся несколько строк с информацией о загрузке шрифтов:

```
(C:\texmf\tex\latex\cyrillic\t2acmr.fd) (C:\texmf\tex\latex\cyrillic\t2acmtt.fd
)
```

Существенно более подробная информация о шрифтах записывается в протокол компиляции, т. е. файл `second.log`, который можно изучить после завершения работы программы `latex`.

После следующего сообщения выполнение программы приостанавливается, так как она встретила ошибку.

```
! LaTeX Error: Environment eqution undefined.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.50 \begin{eqution}
                     % пропущена буква ‘a’ в слове equation
?
```

Первая строка этого сообщения начинается с восклицательного знака и называется *строкой индикации ошибки*. Она содержит описание ошибки. В данном примере ответ прост — процедуры с именем `eqution` не существует. Тем не менее в следующих строках сообщения предлагается для объяснения причины ошибки обратиться к учебникам по LaTeX'у или нажать клавишу ⟨H⟩ и затем ⟨Return⟩ (или ⟨Enter⟩), чтобы получить подсказку на экране.

Предпоследние две строки сообщения называют *строками локализации ошибки*, причём первая заканчивается ошибочной командой `\begin{eqution}`. Перед

этой командой указывается номер строки 50 во входном файле, где эта ошибка находится. Признаком номера строки (по-английски: line) служит буква `l` с точкой на конце. Вслед за строкой локализации ошибки `latex` выводит знак вопроса в начале новой строки и ждёт реакции пользователя. В ответ можно либо нажать клавишу ⟨Enter⟩, чтобы проигнорировать ошибку, либо ввести букву `H`[10], чтобы получить рекомендацию относительно дальнейших действий (на английском языке!). Другие возможные варианты: ввести букву `X`, чтобы немедленно прекратить исполнение программы; ввести букву `I`, чтобы затем набрать исправленный вариант команды (но входной файл потом всё равно придётся исправлять); ввести `Q` или `R`, чтобы компилятор больше не останавливался, встретив новые ошибки. В любом случае информация о всех ошибках будет записана в файл `second.log`. Его можно использовать для последующего анализа ошибок. Нужно только иметь в виду, что одна действительно существующая ошибка часто вызывает несколько сообщений о наведённых ошибках. Именно так обстоит дело в нашем примере: из-за одной пропущенной буквы в строке 50 компилятор зафиксирует 4 ошибки в строках 51 и 52, хотя там их нет. Наведённые ошибки можно смело игнорировать, нажимая клавишу ⟨Enter⟩.

В приложении B приведено описание многих ошибок и способов их устранения.

1.14. Ходит информация по кругу

Входной файл является всего лишь вершиной айсберга информации, которую перерабатывает LaTeX. Рис. 1.2 показывает потоки данных при работе компилятора `latex`. Большие тексты, такие как наша книга, удобно разбивать на несколько файлов. Например, исходный текст каждой главы может составлять отдельный файл. Однако всегда какой-то файл является главным, или *корневым*. Именно с него начинается компиляция документа. На рисунке он назван *jobname*`.tex`. Корневой файл может состоять из команд `\input{`*chapter*`}` или `\include{`*chapter*`}`, каждая из которых как бы дословно переписывает содержание файла *chapter*`.tex` в корневой файл. Дробление больших текстов на небольшие части имеет много достоинств, которые мы обсудим в разделе 3.8.

Информацию, необходимую для своей работы, `latex` получает, считывая форматный файл с расширением `fmt`, содержащий в бинарном виде команды, составляющие *формат* («ядро») LaTeX'а, файлы с описанием класса (они имеют расширение `cls`) и пакетов (`sty`). Классы и пакеты, в свою очередь, могут инициировать загрузку дополнительных файлов, например `clo` или `def`.

Информацию о шрифтах `latex` получает из файлов определения шрифтов, имеющих расширение `fd`. Они сопоставляют декларации переключения шрифтов (раздел 1.11) конкретным шрифтам. Далее `latex` загружает *метрические* файлы шрифтов с расширением `tfm`, где записаны ширина и высота всех литер. Собственно шрифты, т. е. изображения литер, программа `latex` не использует.

[10] Подразумевается, что ввод буквы завершается нажатием клавиши ⟨Enter⟩.

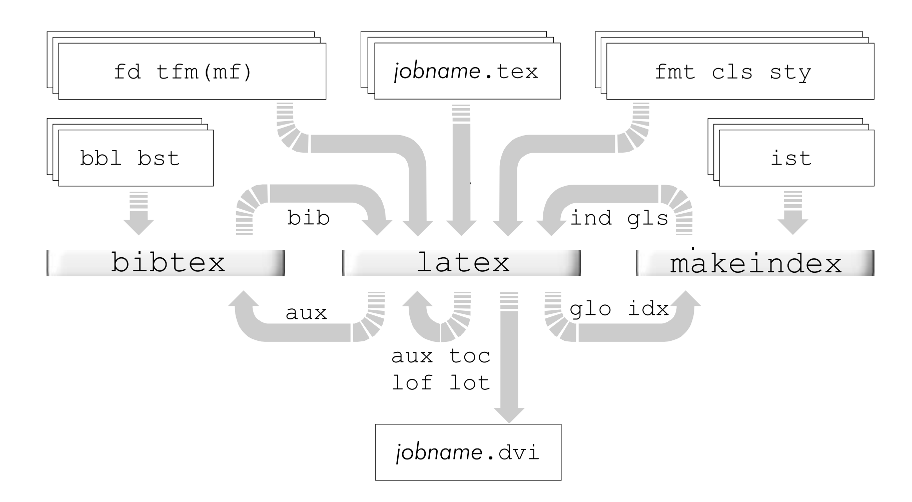


Рис. 1.2. Потоки информации при работе компилятора `latex`

При отсутствии какого-либо файла метрики `tfm` он автоматически будет создан из файла `mf`, где записана программа генерации шрифта.

Программа `latex` считывает информацию также из служебных файлов, которые сама же создаёт при каждой компиляции документа.

Файлы с расширением `aux`, главный из которых имеет имя *jobname*`.aux`, содержат информацию, необходимую для перекрёстного цитирования формул, таблиц, рисунков и вообще любых объектов, имеющих номер.

В файл *jobname*`.toc` записывается информация для формирования оглавления; он создаётся, если исходный текст содержит команду `\tableofcontents`. Файлы *jobname*`.lot` и *jobname*`.lof` соответственно содержат список таблиц и рисунков, размещаемых процедурами `table` и `figure`. Эти списки печатаются командами `\listoftables` и `\listoffigures` (раздел 3.10).

Файл *jobname*`.bib` используется для автоматической нумерации ссылок на цитируемую в тексте литературу из библиографической базы данных, содержащейся в файлах с расширением `bbl`. Работу с библиографической базой данных выполняет программа `bibtex`, распространяемая в составе системы LaTeX. Однако список литературы можно составить и без программы `bibtex`; его печатает процедура `thebibliography`. В любом варианте `latex` автоматически перенумерует все ссылки при внесении изменений в текст документа (глава 13).

Файл *jobname*`.idx` содержит список *входов* в алфавитный указатель. Он является входным для программы `makeindex`, которая записывает отсортированный список в файл *jobname*`.ind`. Команда `\printindex` вставляет его в печатный документ в виде полностью сформатированного алфавитного указателя. Файл *jobname*`.glo` содержит список *входов* в глоссарий (словарь терминов). Его также можно отсортировать с помощью программы `makeindex`, которая запишет результат в файл *jobname*`.gls` (глава 14).

Программы `bibtex` и `makeindex` могут загружать дополнительную информацию для своей работы — так называемые файлы стилей с расширениями `bst` и `ist`, которые являются аналогами пакетных файлов `sty`, используемых программой `latex`.

Компилятор `latex` производит несколько файлов.

Файл *jobname*`.dvi` содержит смакетированный текст печатного документа в формате DVI. Название формата происходит от слов «DeVice Independent», что должно было бы означать независимость изображения документа от разрешения печатного устройства. На деле же, конечно, качественное изображение невозможно получить на выходном устройстве с малым разрешением, но чем выше разрешение устройства, тем выше качество изображения, причём один и тот же документ DVI одинаково пригоден для воспроизведения на совершенно разных устройствах, поскольку не содержит растровых изображений. Например, в `dvi`-файле записаны координаты *литер* на каждой странице, но не сами изображения букв. Предполагается, что литеры будут подставлены *обозревателем* документов DVI непосредственно перед показом документа на экране или при печати на принтере сообразно разрешению выходного устройства.

Файл *jobname*`.log` (не показан на рисунке) содержит протокол работы компилятора, в том числе всю информацию, которая появляется на экране: загруженные файлы, номера страниц, сообщения об ошибках и т. п.

При каждой новой компиляции исходного файла `latex` обновляет информацию, записанную в служебные файлы, и затем проверяет, совпадают ли новые версии служебных файлов со старыми. В случае несовпадения `latex` печатает предупреждение:

```
LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right[11].
```

Не найденные перекрёстные ссылки изображаются в печатном документе двумя вопросительными знаками. Если в текст добавлена новая глава, то при первой компиляции информация о ней будет только записана в файл *jobname*`.toc`, и лишь при следующем проходе она будет включена в оглавление. Однако на завершающей стадии работы над текстом в него вносится обычно очень незначительная правка, которая не меняет содержание служебных файлов. Поэтому при последней компиляции проблем, как правило, не возникает. Если же компилятор всё-таки выдал предупреждение, достаточно пропустить входной файл через `latex` ещё раз.

Аналогичным образом работает компилятор `pdflatex`. Он создаёт из исходного текста печатный документ формата PDF. Как видно из сравнения рисунков 1.2 и 1.3, существенное отличие состоит в том, что `pdflatex` загружает шрифты и внедряет их в выходной файл *jobname*`.pdf`. Программа `pdflatex` может внедрить в документ `pdf` как растровые шрифты `pk` (которые при необходимости автоматически генерируются из `mf`-файлов), так и векторные шрифты PostScript

[11] Предупреждение LaTeX'а: Возможно, метки изменились. Запустите ещё раз для получения правильных ссылок.

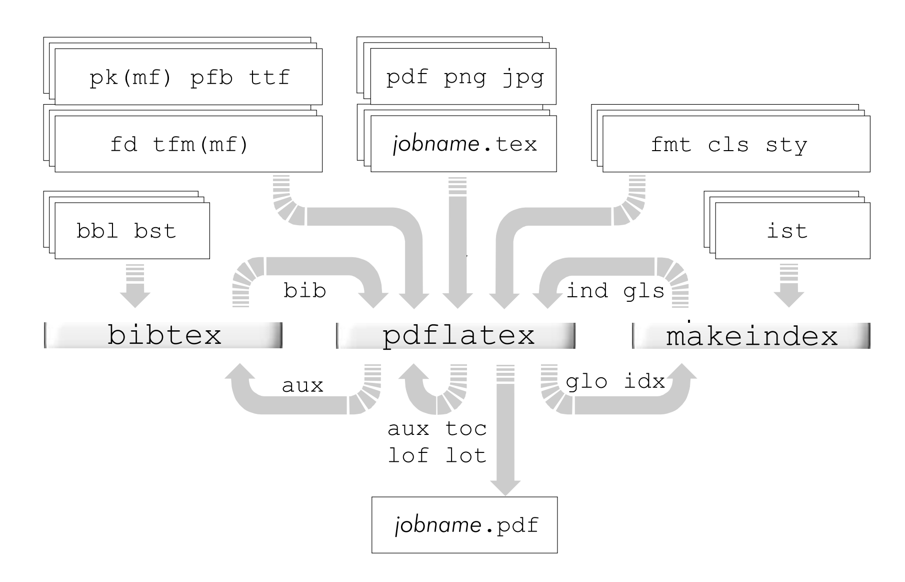


Рис. 1.3. Потоки информации при работе компилятора `pdflatex`

(`pfb`), TrueType и OpenType (`ttf`). Чтобы уменьшить размер документа PDF, можно отключить внедрение векторных шрифтов, если есть уверенность, что на компьютере получателя документа имеются все необходимые шрифты.

Ещё одно отличие состоит в том, что `pdflatex` внедряет в выходной файл *jobname*`.pdf` рисунки. На рис. 1.3 перечислены форматы графических изображений, с которыми `pdflatex` умел обращаться на момент издания нашей книги: `pdf`, `png` и `jpg`.

Откомпилированный документ можно напечатать на принтере или просмотреть на экране монитора. Способы визуализации файлов `dvi` и `pdf` представлены на рисунках 1.4 и 1.5 соответственно.

Печать и вывод на экран дисплея документов DVI осуществляют специальные программы — DVI-обозреватели. Эти программы на основе информации в файле *jobname*`.dvi` создают растровое изображение печатного документа, состоящее из *пикселей* (точек разных цветов). Полученное изображение выводится на устройство с заданным *разрешением*. Разрешение устройства характеризуется числом пикселей на единицу длины в один дюйм. В комплект программ MiKTeX входит DVI-обозреватель `YAP`. Схема его работы представлена на рис. 1.4. Примерно так же работают другие современные DVI-обозреватели. Поскольку в файл `dvi` не внедрены шрифты и рисунки, они должны присутствовать на жёстком диске компьютера в момент работы `YAP`. К достоинствам `YAP` относится его способность использовать векторные шрифты PostScript (`pfb`), а также TrueType и OpenType (`ttf`). Программа `YAP` автоматически генерирует растровые шрифты (`pk`) из векторных шрифтов (`mf`, `pfb`, `ttf`). Она также имеет средства *обратного поиска*: при двойном клике указателем мышки в какой-нибудь части окна `YAP` происхо-

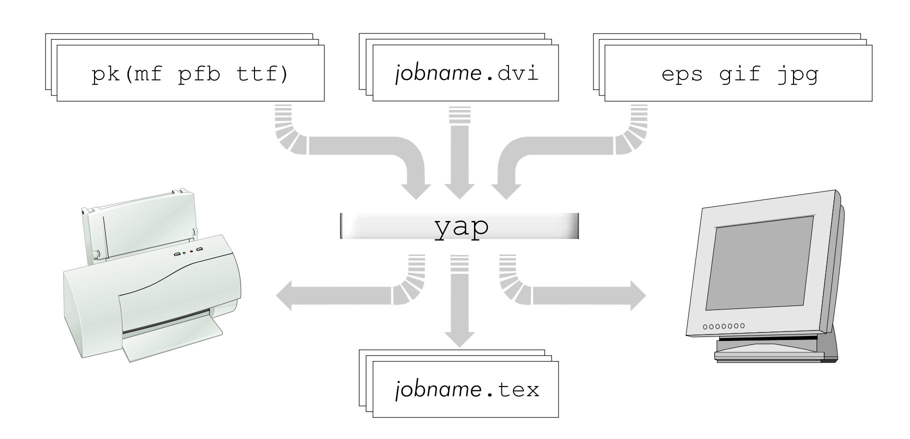


Рис. 1.4. Визуализация документа DVI

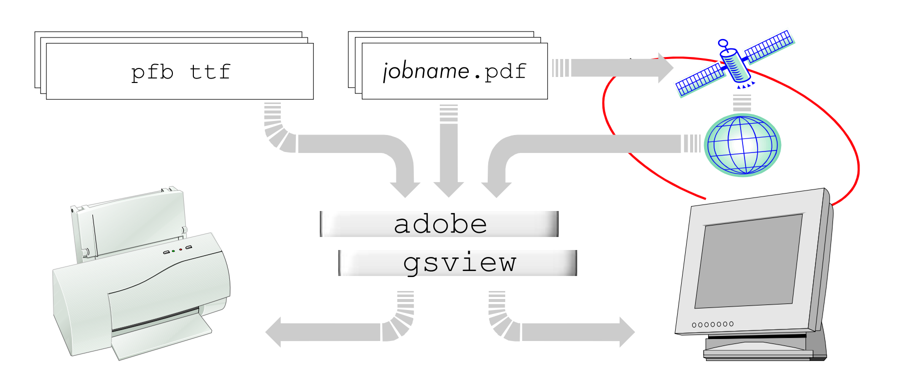


Рис. 1.5. Визуализация документа PDF

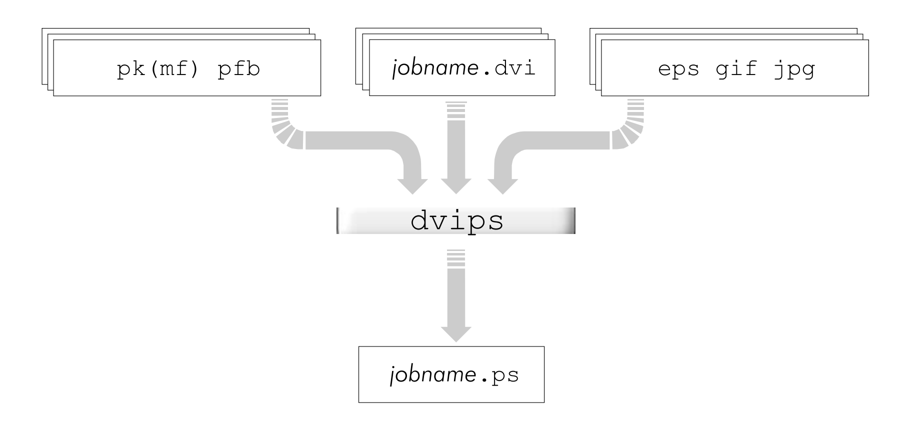


Рис. 1.6. Преобразование документа DVI в PostScript

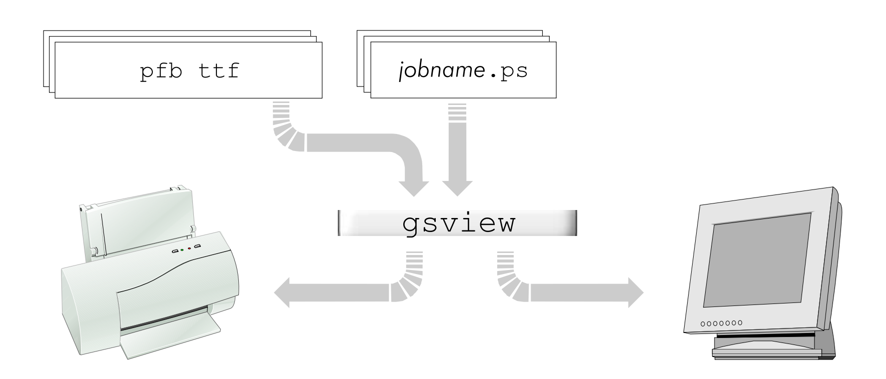


Рис. 1.7. Визуализация документа PostScript

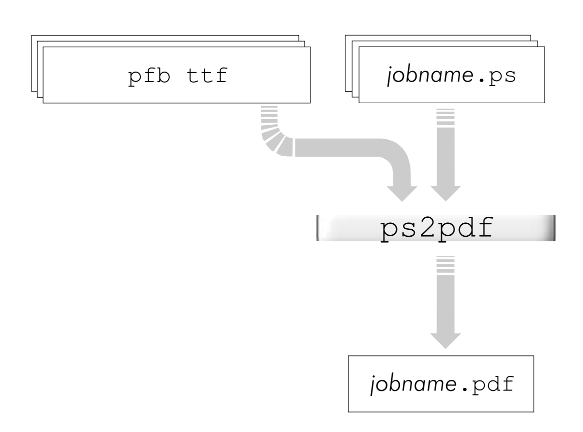


Рис. 1.8. Преобразование документа PostScript в PDF

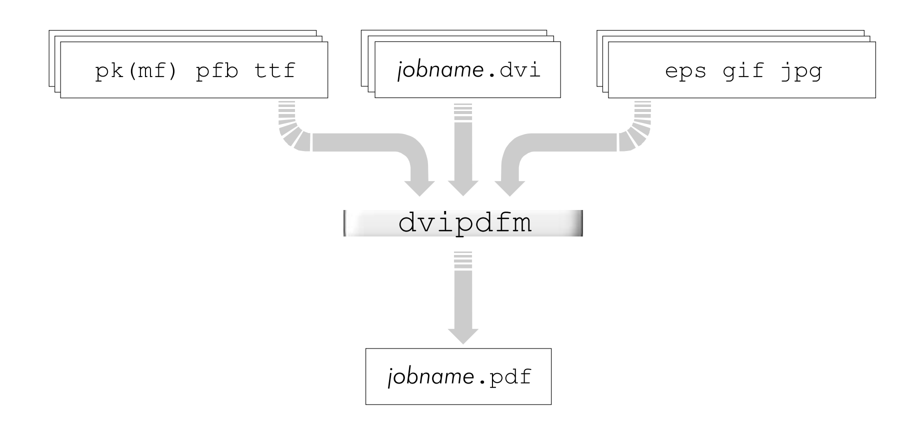


Рис. 1.9. Преобразование документа DVI в PDF

дит переход к соответствующему исходному тексту в окне редактора. Печать и вывод на экран дисплея документов PDF выполняет программа Adobe Reader[12], бесплатно распространяемая фирмой Adobe. Имеется также более изощренная коммерческая версия этой программы, известная под названием Adobe Acrobat. На рис. 1.5 Adobe Reader и Adobe Acrobat обозначены одним словом `adobe`. Вследствие доступности программы Adobe Reader документ PDF легко переносить с одного компьютера на другой и даже экспонировать на Web сайтах, поскольку Adobe Reader легко встраивается в Web браузеры. Если в документ PDF внедрены не все необходимые шрифты, Adobe Reader может их догружать в момент открытия документа с жёсткого диска компьютера получателя документа (конечно, если они там имеются).

Вместо Adobe Reader можно использовать программу GSview. Она представляет собой графический интерфейс к библиотеке программ Ghostscript. Эта библиотека поставляется отдельно от системы LaTeX, но де-факто является её неотъемлемой составной частью. Многие DVI-обозреватели используют Ghostscript для показа на экране дисплея и печати рисунков PostScript.

Язык описания страниц PostScript является предшественником формата PDF и также разработан фирмой Adobe. Преобразование файла `dvi` в PostScript-файл (с расширением `ps`) производит программа `dvips` согласно схеме на рис. 1.6. Интерпретаторы языка PostScript встроены во многие принтеры, поэтому `ps`-файл достаточно скопировать («отправить») на такой принтер, чтобы напечатать.

Если выходное устройство (например, экран монитора) не имеет встроенного интерпретатора, документ PostScript можно просмотреть и напечатать с помощью программы GSview, как показано на рис. 1.7. В настоящее время необходимость в преобразовании документа LaTeX в PostScript возникает редко. В качестве примера можно упомянуть разве что случай, когда требуется разместить на одном листе несколько страниц документа для последующей печати в виде буклета, получить зеркальное или негативное изображение. Поскольку PostScript по сути дела является языком программирования, все такие операции легко выполнить программным образом (раздел 17.5).

Многие пользователи привыкли к программе `dvips`, которая оказала существенное влияние на эволюцию системы LaTeX. Мы упомянули эту программу главным образом для того, чтобы упростить таким пользователям переход к современным способам компиляции документов LaTeX.

Рисунки 1.8 и 1.9 иллюстрируют ещё два варианта преобразования файлов `ps` и `dvi` в `pdf`, в значительной мере утративших свою актуальность. PostScript-файл можно преобразовать в `pdf` при помощи программ Ghostscript или Adobe Distiller. Соответствующий пункт меню редактора WinEdt называется `ps2pdf` (рис. 1.8). Для получения `pdf` из `dvi`-файла следует использовать программу `dvipdfm` (рис. 1.9). Мы не будем здесь детально обсуждать, когда имело смысл использовать подобные многоступенчатые способы получения PDF документов, а лишь посоветуем Читателю обратиться за пояснениями к главе 10.

[12] До версии 5 включительно она называлась Acrobat Reader.

Да закройте же скобки, мужчина!
Вл. Вишневский. Одностишья

Глава 2

Команды и процедуры

Общие правила синтаксиса LaTeX'а объясним на примере команды

`\documentclass[`*options*`]{`*class*`}[`*release-date*`]`

Условимся, что текст, набранный машинописным шрифтом, как `\documentclass`, квадратные и фигурные скобки, следует вводить во входной файл в буквальном соответствии с определением команды. Напротив, текст, набранный прямым курсивом: *options*, *class*, *release-date*,— может изменяться. В данном случае `\documentclass` — это *имя* команды, а *options*, *class* и *release-date* — *аргументы* команды. Аргумент в фигурных скобках является *обязательным*. Он иногда может быть пустым — `{}` (даже без пробела между скобками), но пропуск самих фигурных скобок обычно[1] приводит к ошибке. Аргументы в квадратных скобках `[ ]` *не обязательны*. Их часто называют *опциями*. Все или некоторые опции (вместе с квадратными скобками) могут быть опущены, так что краткая форма команды `\documentclass` такова:

`\documentclass{`*class*`}`

Пропущенные необязательные аргументы принимают значения *по умолчанию*. Если синтаксис команды допускает два необязательных аргумента, идущих один за другим, а использован только один, предполагается, что указан первый, а опущен второй. Всё не существенное в текущем контексте будем заменять многоточием. Так, если бы не было необходимости объяснять смысл обозначения *class*, то приведённый выше пример мог быть таким:

`\documentclass{`...`}`

Определения команд мы будем заключать в рамку в отличие от примеров их применения. Наиболее важные упоминания той или иной команды в книге можно отыскать при помощи алфавитного указателя, причём номера страниц с определениями команд и процедур выделены в указателе курсивом.

Пробелы между аргументами команд, а также между именем команды и первым аргументом игнорируются.

[1] Аргумент команды можно не заключать в фигурные скобки, если он состоит из одного символа или одной команды, не имеющей собственных аргументов.

2.1. Имя команды

Шесть команд:

```
# $ & ~ _ ^
```

имеют имя, состоящее только из одного символа. Символ `%` не считается командой, но указывает, что компилятор должен игнорировать всё, что следует за ним до конца строки; `%` используется, чтобы отделить комментарий или начать новую строку без ввода пробелов в конце предыдущей. Например:

`абрака% Это комментарий` `дабра`	абракадабра

Все другие команды обычно начинаются с символа \ (*обратный слеш*), причём одна команда состоит только из этого символа, за которым следует пробел[2]. Поскольку для обозначения обязательного пробела принято использовать символ ␣, эта команда записывается в виде \␣.

Особую группу образуют команды, которые состоят из \ и одного символа, не являющегося буквой. Читателю уже известно, что команда `\,` (обратный слеш с запятой) добавляет пробел и может использоваться для раздвижения кавычек (здесь можно попробовать отыскать при помощи алфавитного указателя соответствующий пример в первой главе). Ещё одна команда `\\` из этой группы встретилась нам в строке 56 в примере входного файла в разделе 1.3. Она вызывает принудительный переход на новую строку в печатном документе.

В именах остальных команд за символом \ следует одна или несколько букв латинского алфавита. Признаком конца имени такой команды служит любой символ, не являющийся латинской буквой, а именно: цифра, знак препинания[3], знаки математических операций (кроме `*`), пробелы, служебные символы, а также русские буквы. Все специальные символы, которые могут быть признаком конца имени команды, перечислены в разделе 1.4. Имя команды нельзя разбить на части для переноса между соседними строками даже с помощью `%`.

При компиляции исходного текста буквы русского алфавита транслируются в команды `\cyr`*cmd* или `\CYR`*cmd*. Это подтверждает следующий пример:

`\CYRR\cyru\cyrs\cyrs\cyrk\cyri\cyrishrt\` `\CYRYA\cyrz\cyrery\cyrk`	Русский Язык

Исходный текст на русском языке, набранный в командах `\cyr`*cmd* и `\CYR`*cmd*, будет правильно откомпилирован без перекодировки системой LaTeX, работающей на любой платформе. В явном виде команды `\cyr`*cmd* и `\CYR`*cmd* используются только внутри пакетов — так обеспечивается переносимость пакетов между компьютерами разных типов. Трансляцию русских букв в команды в процессе компиляции исходного текста производит пакет inputenc. Поскольку русские

[2] Пакет babel вводит команды, которые начинаются с символа двойных кавычек `"`, но мы не рекомендуем использовать такие команды.

[3] Команды с именами, содержащими символ `@`, относящийся к знакам препинания, могут использоваться в служебных файлах с расширениями `cls`, `clo`, `sty`.

буквы по сути являются командами, их нельзя использовать в именах других команд. Любая русская буква будет воспринята компилятором в качестве признака конца имени команды.

Чаще всего за именем команды следует левая фигурная или квадратная скобка, открывающая аргумент, или пробел, если аргументов нет. Пробелы после имени команды игнорируются. LaTeX различает строчные и прописные буквы, поэтому `\Omega` и `\omega` есть прописная (Ω) и строчная (ω) буквы греческого алфавита.

Некоторые команды имеют особую `*`-форму, когда за именем команды стоит знак `*` («звёздочка»). Пример такой команды имеется в следующем разделе.

2.2. Аргументы

Некоторые команды имеют только один и только необязательный аргумент. С одной такой командой Читатель уже знаком, хотя мы и не упоминали, что у `\\` может быть аргумент. Если команда `\\` просто начинает новую строку, то `\\[2mm]` ещё и увеличивает расстояние до новой строки на 2 миллиметра:

```
альфа\\[2mm] бета\\ гамма
```

альфа

бета
гамма

Если необязательный аргумент у такой команды опущен, а следующий отличный от пробела символ есть `[`, то LaTeX ошибочно примет эту квадратную скобку за начало необязательного аргумента. Чтобы избежать этой редкой ошибки, символ `[` достаточно заключить в фигурные скобки. В следующем примере слово `[2mm]` не является больше частью команды `\\` и поэтому напечатано во второй строке правой колонки, показывающей результат работы компилятора:

```
альфа\\{[2mm]} бета\\ гамма
```

альфа
[2mm] бета
гамма

Среди команд, обладающих `*`-формой, некоторые (в их числе всё та же команда `\\`) не имеют обязательных аргументов. Чтобы отличить команду `\\`, за которой следует текст, начинающийся со `*`, от команды `\\*`, звёздочку нужно заключить в фигурные скобки:

```
альфа\\* бета\\{*} гамма
```

альфа
бета
* гамма

В этом примере команда `\\*`, как и `\\`, создаёт новую строку, однако препятствует её переносу на следующую страницу (раздел 4.4). Поэтому первая звёздочка в левой колонке является частью имени команды `\\*`, а вторая — частью текста, следующего за `\\`.

2.3. Командные скобки и процедуры

Процедура[4] начинается с командной скобки `\begin` и закрывается командной скобкой `\end`:

```
\begin{env} ... \end{env}
```

Многоточие здесь символизирует *тело процедуры*. Аргумент *env* есть *имя процедуры*. Команда `\begin` может иметь дополнительные аргументы. Например, процедура `tabular` описана в главе 12 как

```
\begin{tabular}[pos]{col} ... \end{tabular}
```

Она имеет необязательный аргумент *pos* и обязательный *col*. Если процедура имеет `*`-форму, звёздочка присоединяется к имени процедуры, а не к команде `\begin`. Например:

```
\begin{table*} ... \end{table*}
```

где `table*` — процедура, описанная в главе 11.

Процедуры могут быть вложены друг в друга. Иными словами, каждой открывающей командной скобке `\begin{env1}` должна соответствовать закрывающая командная скобка `\end{env1}`, и если внутри процедуры *env1* имеется другая процедура *env2*, то её тело должно целиком находиться внутри тела первой процедуры. Следовательно, чередование командных скобок должно быть следующим:

```
\begin{env1}
... \begin{env2} ... \end{env2} ...
\end{env1}
```

2.4. Группирование

Всякий раз, когда фрагмент исходного текста рассматривается как единый блок, нужно пометить его начало и конец. Для этой цели LaTeX резервирует два символа группирования — фигурные скобки: `{` и `}`. Фигурным скобкам LaTeX отводит особую роль. Можно было бы сказать, что фигурные скобки — это команды, но, строго говоря, это не совсем верно (см. главу 7 в [2]). Знаток TeX'а мог бы назначить другие символы на роль фигурных скобок, но последние вполне справляются со своей задачей.

Читатель уже имел возможность видеть во вводной главе, что изменение шрифта внутри блока не влияет на шрифт за его пределами. Позже мы покажем,

---

[4] В книгах [12–14] употребляется термин «окружение», являющийся буквальным переводом слова environment, используемого в англоязычной литературе. В компьютерной литературе закрепился термин environment variable (переменная окружения), который имеет совершенно иной смысл, нежели термин procedure (процедура), более всего соответствующий понятию environment в языке разметки LaTeX.

что если определить новую команду внутри какого-нибудь блока, то её определение исчезает там, где блок заканчивается. Таким способом можно заставить LaTeX сделать что-то необычное, заменив его стандартные установки внутри блока. Поскольку определение невидимо снаружи, не нужно беспокоиться, что оно изменит остальную часть текста.

Иногда, даже когда не нужно заботиться о структуре блока, полезно применять группирование просто для улучшения контроля за пробелами. Рассмотрим, к примеру, команду `\LaTeX`, которая печатает логос LaTeX. Мы отмечали в разделе 1.5.3, что пробел после команды-логоса исчезает, так как является всего лишь признаком конца команды. Чтобы вставить неудаляемый пробел, предлагалось использовать команду `\␣` после логоса: `\LaTeX\␣`. Однако неверным будет выражение `\LaTeX\`, если следующий символ не является пробелом. Если Читатель попробует написать, что он уже знаток `\LaTeX\a`, то получит от LaTeX'а «неуд». Верное решение даёт простой блок:

```
{\LaTeX}
```

Тогда не имеет значения, будет следующий символ пробелом или нет. Другой простой способ — добавить пустой блок после команды:

```
\LaTeX{}
```

Пустой блок `{}` (даже без пробела внутри) не производит никакого печатного текста, но он предотвращает удаление пробелов, выполняя роль признака конца команды.

2.5. Декларации

Команда, которая изменяет значение или смысл параметра или другой команды, является *декларацией*. В отличие от обычной команды декларация ничего не производит в печатном документе и поэтому может располагаться в преамбуле входного файла. *Область действия* декларации начинается от места её появления и заканчивается закрывающей фигурной скобкой `}`, которая соответствует ближайшей открывающей скобке `{` перед декларацией. Декларация может быть отменена другой декларацией, действие которой противоречит действию первой. В следующем примере `\itshape`, `\scshape` и `\slshape` декларируют переход к шрифтам с разным начертанием, не меняя их насыщенность, гарнитуру или размер:

```
text1 {text2 \itshape text3
{text4} \scshape text5
{\slshape text6}  text7} text8
```

text1 text2 *text3 text4* TEXT5 *text6* TEXT7 text8

Декларация `\itshape` отменяется декларацией `\scshape`, а та — декларацией `\slshape`, которая, в свою очередь, действует только до ближайшей закрывающей фигурной скобки.

Командные скобки \begin{*env*} и \end{*env*} ограничивают область действия деклараций так же, как это делают фигурные скобки. В следующем примере декларация \em действует только внутри тела процедуры quote, описанной в разделе 5.2:

```
Этот текст предшествует
процедуре \texttt{quote}:
\begin{quote}
\em Это тело процедуры.
\end{quote}  Этот текст следует
за процедурой.
```

Этот текст предшествует процедуре quote:

> *Это тело процедуры.*

Этот текст следует за процедурой.

Декларации, находящиеся в аргументах команд LaTeX'а, действуют только в пределах этих аргументов. Иными словами, скобки } и], закрывающие аргументы команд, закрывают и область действия деклараций. Однако это не относится к командам, созданным самим Читателем при помощи \newcommand и \renewcommand (глава 7). При необходимости область действия деклараций в аргументах таких команд следует ограничивать дополнительными фигурными или командными скобками.

Следующие декларации, описанные в разных частях этой книги, являются глобальными:

```
\newcounter      \pagenumbering   \newlength
\setcounter      \thispagestyle   \newsavebox
\addtocounter    \pagecolor       \newtheorem
\hyphenation
```

Область их действия не ограничивается никакими фигурными или командными скобками.

2.6. Невидимые команды

Некоторые команды и процедуры не производят никакого текста в том месте, где они стоят. LaTeX рассматривает такую «невидимую» команду в середине абзаца как невидимый текст.

Вставка пробела или признака конца строки одновременно до и после невидимого слова может создать удвоенный пробел между видимыми словами. Так, в примере на странице 30 между словами «используют *курсив*» образовался удвоенный пробел из-за наличия пробелов до и после команды \begin{em}. В данном случае пробел после {em} не является признаком конца имени команды (оно заканчивается фигурной скобкой) и поэтому не удаляется LaTeX'ом. Для команд без аргументов проблемы с пробелами возникают редко, так как следующий за именем команды пробел игнорируется. Перечисленные ниже невидимые команды и процедуры, имеющие аргументы, также удаляют следующий за ними пробел, если имеется пробел перед ними:

```
\pagebreak     \linebreak     \vspace      \color
\nopagebreak   \nolinebreak   \marginpar   figure
\label         \index         \glossary    table
```

Любые другие невидимые команды с аргументами, появляющиеся внутри абзаца, должны быть присоединены непосредственно к предыдущему слову. Полезно следовать этому правилу всегда, чтобы не задумываться о возможных нестандартных ситуациях, когда вышеперечисленные команды и процедуры могут удвоить пробел. В сложных случаях можно использовать команду

`\ignorespaces`

Она удаляет все пробелы, следующие за ней:

```
абра\ignorespaces   кадабра
```

абракадабра

2.7. Хрупкие команды

В первой главе мы уже отмечали, что LaTeX может переносить аргументы одних команд в аргументы других команд. Аргументы, которые куда-либо передаются, называются *подвижными*. Над командой, находящейся в подвижном аргументе, LaTeX проделывает ряд операций, которые могут расстроить механизм её работы. Команды, поддающиеся расстройке, называются *хрупкими* в отличие от *устойчивых* команд, которым ничего не страшно. Например, команды `\begin`, `\end`, `\footnote` являются хрупкими, а декларации и команды переключения шрифта типа `\Large`, `\bfseries`, `\textbf` — устойчивыми.

Хрупкой команде, помещённой в подвижном аргументе, должна предшествовать команда

`\protect`

Она действует только на следующую за ней команду, которую не нужно заключать в фигурные скобки:

```
\protect\begin ...
```

Хороший пример с `\protect` придумать трудно, так как обычно хрупким командам нечего делать в подвижных аргументах других команд. Все нижеследующие команды и процедуры имеют подвижные аргументы.

- Команды, аргументы которых могут быть вставлены в оглавление, список рисунков и таблиц, как-то: команды секционирования типа `\chapter`, `\section` и т. д. (раздел 3.5); команды `\addcontentsline` и `\addtocontents`, записывающие информацию в служебные файлы (раздел 3.10); `\caption`, формирующая подписи к рисункам и таблицам (глава 11). Если необязательный аргумент используется с `\caption` или с командами секционирования, то именно он является подвижным.

- Команды вывода на терминал `\typeout` и ввода с клавиатуры `\typein` (раздел 3.9). Необязательный аргумент `\typein` не является подвижным.
- Команды, генерирующие текст верхнего колонтитула страницы: `\markboth` (оба аргумента) и `\markright` (раздел 17.1).
- Процедура `letter` для составления писем (раздел 15.4).
- Команда `\thanks` для составления подстрочного примечания на титульной странице печатного документа (раздел 3.4).
- Символ `@` в процедурах `array` и `tabular` (глава 12). Хотя `@` не является командой, хрупкие команды в выражениях с `@` должны быть защищены посредством `\protect`, как если бы они находились в подвижном аргументе.

Определения хрупких команд в дальнейшем снабжены значком ⚠ слева от команды. Время от времени разработчики LaTeX 2_ε переводят команды, бывшие хрупкими, в разряд устойчивых. Поэтому некоторые из значков ⚠ в нашей книге могут оказаться излишними. Однако команда `\protect` безвредна и может предшествовать без всяких последствий даже устойчивой команде. Единственное исключение состоит в том, что команда `\protect` не должна стоять перед командами, имеющими смысл длины (раздел 2.10), и не должна использоваться в аргументах команд `\setcounter` и `\addtocounter`, работающих со счётчиками (раздел 2.9).

2.8. Режимы форматирования

Обрабатывая исходный текст, LaTeX всегда находится в одном из четырёх режимов:

1) текстовый режим (paragraph mode);
2) строковый режим (LR mode);
3) математический режим (math mode);
4) графический режим (restricted LR mode).

В текстовом режиме LaTeX рассматривает исходный текст как последовательность слов, которые нужно разбить на строки, абзацы и страницы. Это основной режим форматирования. LaTeX первоначально всегда находится в текстовом режиме.

Когда LaTeX переходит в строковый режим (LR — от слов Left to Right), он не разбивает текст на строки вне зависимости от их длины. В этом режиме LaTeX обрабатывает аргументы таких команд, как `\mbox{`*text*`}`, `\fbox{`*text*`}` (раздел 9.1).

В математический режим LaTeX переходит, когда обрабатывает тело процедур, описанных в главе 6, посвящённой математическим формулам.

Графический режим включается, когда LaTeX обрабатывает тело процедуры `picture`. Ей посвящена часть главы 9.

Режимы форматирования могут быть вложены друг в друга так же, как команды или процедуры. Математическую формулу можно включить в рисунок, а рисунок — в формулу. Примеры подобных вложений мы встретим не раз.

2.9. Счётчики

LaTeX автоматически нумерует страницы, главы, рисунки, уравнения и многое другое. Каждому числу, которое генерирует LaTeX, соответствует счётчик. Имя счётчика обычно совпадает с именем использующей его процедуры или команды. Ниже приведена таблица основных счётчиков, которые используются стандартными классами LaTeX'а:

`part`	`paragraph`	`figure`	`enumi`
`chapter`	`subparagraph`	`table`	`enumii`
`section`	`page`	`footnote`	`enumiii`
`subsection`	`equation`	`mpfootnote`	`enumiv`
`subsubsection`			

Первые семь счётчиков (от `part` до `subparagraph`) в двух первых колонках используются для нумерации разделов печатного документа; их имена совпадают с именами команд секционирования (раздел 3.5), но не содержат `\`. Номер текущей страницы хранится в счётчике `page`, номер последней пронумерованной формулы записан в счётчике `equation`, который используется процедурами `equation` и `eqnarray` (глава 6) и ещё десятком других (глава 8). В третьей колонке собраны счётчики, используемые для нумерации рисунков, таблиц (глава 11), подстрочных примечаний (раздел 4.8) и подстрочных примечаний в министраницах, изготавливаемых процедурой `minipage` (глава 9). В последней колонке сгруппированы счётчики, контролирующие нумерацию записей в пронумерованном списке процедуры `enumerate` (раздел 5.4.2). Имеется ещё ряд счётчиков, контролирующих некоторые параметры настройки печатного документа; их описание разбросано по многим главам книги. Ниже мы покажем, как в дополнение к имеющимся можно определять новые счётчики.

Значение счётчика — целое число, обычно неотрицательное. Следует отличать значение счётчика ctr от формы представления этого значения командой `\the`ctr. Например, номер текущей главы хранится в счётчике `chapter`, а печатает её номер команда `\thechapter`:

```
Это пример в главе \thechapter.
```
Это пример в главе 2.

Значение любого счётчика ctr может быть напечатано арабскими или римскими цифрами, буквами или знаками сноски в зависимости от того, как определена команда

`\the`ctr

где ctr — имя счётчика. Составной номер, который Читатель видит в заголовках разделов, генерируется несколькими счётчиками. Номер данного раздела складывается из номера главы 2 и номера секции 9.

Изменить формат представления `\thectr`, установленный классом печатного документа, можно при помощи следующих команд:

`\arabic{ctr}`	`\fnsymbol{ctr}`
`\roman{ctr}`	`\Roman{ctr}`
`\alph{ctr}`	`\Alph{ctr}`

Пакет babel с опцией `russian` добавляет к ним ещё две:

`\asbuk{ctr}`	`\Asbuk{ctr}`	(babel)

Значения счётчика ctr «один», «два», «три» команда `\arabic` напечатает арабскими цифрами 1, 2, 3; команды `\roman` — строчными римскими цифрами i, ii, iii; `\Roman` — прописными римскими цифрами I, II, III; `\alph` и `\Alph` — соответственно строчными и прописными буквами латинского алфавита a, b, c, A, B, C; `\asbuk` и `\Asbuk` — русскими буквами а, б, в, А, Б, В; `\fnsymbol` — подстрочными символами $*$, †, ‡. Всего имеется 9 подстрочных символов: $*$ † ‡ § ¶ ‖ $**$ †† ‡‡. Если значение счётчика печатается буквами, то оно не должно быть больше числа букв в используемом алфавите и уж во всяком случае должно быть положительным.

```
Напечатаем номер текущей главы
римскими цифрами \Roman{chapter}
и напечатаем значение счётчика
(\thechapter).
```

Напечатаем номер текущей главы римскими цифрами II и напечатаем значение счётчика (2).

Предположим, что главы (chapter) необходимо пронумеровать прописными римскими цифрами, а секции (section) — буквами, причём номер секции должен включать в себя номер главы. Чтобы реализовать это желание, достаточно переопределить команды `\thechapter` и `\thesection`. Существующие команды переопределяются при помощи декларации `\renewcommand`. Она во всех деталях описана в разделе 7.1, но следующий пример столь прост, что не нуждается в комментариях:

```
\renewcommand{\thechapter}{\Roman{chapter}}
\renewcommand{\thesection}{\thechapter.\Asbuk{section}}
```

Теперь номер второй секции четвертой главы будет выглядеть как IV.Б.

Многочисленные процедуры и команды, работающие со счётчиками, используют небольшое количество более простых команд, к описанию которых мы сейчас и переходим. Любые команды, изменяющие значение счётчика, имеют *глобальную* область действия, которая не ограничивается никакими фигурными или командными скобками.

Новый счётчик вводится декларацией

```
\newcounter{newctr}[outctr]
```

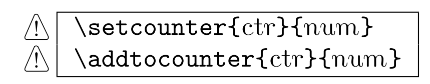
с двумя аргументами *newctr* и *outctr*, второй из которых необязателен.

newctr — имя нового счётчика, составленное из букв. Оно не должно совпадать с именем любого из существующих счётчиков. Новый счётчик инициализируется нулем, а команда `\the`*newctr* определяется так, чтобы значение счётчика печаталось арабскими цифрами, а именно `\arabic{`*newctr*`}`.

outctr — имя уже существующего счётчика. Если этот аргумент присутствует, то счётчик *newctr* объявляется внутренним для счётчика *outctr*. Тогда значение счётчика *newctr* сбрасывается до нуля всякий раз, когда счётчик *outctr* наращивается командами `\stepcounter` или `\refstepcounter` (см. ниже).

Декларация `\newcounter` не может быть использована в файлах, вставляемых командой `\include` (раздел 3.8).

Следующие две декларации изменяют значения существующего счётчика *ctr*:



```
\setcounter{ctr}{num}
\addtocounter{ctr}{num}
```

Первая из них присваивает счётчику *ctr* целое значение *num*. Вторая увеличивает счётчик *ctr* на целую величину *num*. Значение *num* может быть отрицательным. Обе декларации воздействуют только на те счётчики, которые указаны в их аргументах.

Первоначально большинство счётчиков инициализируется нулём. Команды и процедуры, работающие со счётчиками, сначала наращивают их значение, а потом используют полученное значение.

```
\setcounter{footnote}{8}
Счётчик
задан\footnote{Первая сноска.}
\addtocounter{footnote}{-2}
искусственно\footnote{Вторая
сноска.}.
```

Счётчик задан[9] искусственно[8].

[9] Первая сноска.
[8] Вторая сноска.

Счётчик `page`, в котором хранится номер текущей страницы, отличается тем, что наращивается после того, как завершено формирование страницы. Поэтому `page` инициализируется всегда единицей, а не нулем (см. также раздел 17.1). Команда `\setcounter{page}{34}`, помещённая в середину документа, назначит текущей странице номер 34. Последующие номера страниц будут наращиваться от этого числа.

Команда

```
\value{ctr}
```

даёт «чистое» значение счётчика *ctr* (не зависящее от формы его представления `\the`*ctr*). Она используется обычно в аргументе *num* команд `\setcounter` и `\addtocounter`. Например, `\setcounter{footnote}{\value{page}}` приравнивает номер подстрочного примечания к номеру текущей страницы. Однако команда `\value` может использоваться в любом месте, где LaTeX ожидает получить число. Команда `\value` устойчивая, её ни при каких условиях не следует защищать командой `\protect`.

Ещё две команды

```
\stepcounter{ctr}
\refstepcounter{ctr}
```

используются для наращивания значения счётчиков. Однако их функция не сводится к простому увеличению значения *ctr* на единицу. Для этой цели достаточно было бы команды `\addtocounter` со вторым аргументом *num*, равным единице. Помимо увеличения *ctr* на 1, эти две команды также сбрасывают до нуля значения любых внутренних счётчиков. Счётчик *ctr* является внутренним для счётчика *outctr*, если первый был установлен командой `\newcounter` с опцией *outctr* (см. выше). Стандартные стили определяют, что счётчик `subsubsection` является внутренним для `subsection`, который, в свою очередь, является внутренним для `section` и т. д.

Команда `\refstepcounter{`*ctr*`}` имеет ещё одну функцию. Она декларирует, что текст, генерируемый командой `\the`*ctr*, будет текущим `ref`-значением для организации перекрёстного цитирования (раздел 3.7).

В ядре LaTeX'а не предусмотрена декларация для переопределения существующего счётчика по аналогии с `\renewcommand`, хотя иногда возникает желание не только изменить значение счётчика (что легко сделать с помощью `\setcounter`), но и добавить или, наоборот, удалить внешний счётчик для существующего счётчика. Например, объявив счётчик `section` внешним для счётчика `equation`, можно было бы легко организовать независимую нумерацию уравнений в пределах каждого отдельного раздела, хотя стандартные классы документов обычно предусматривают сквозную нумерацию уравнений в пределах всего документа. Пакет amsmath, описанный далее в главе 8, восполняет этот пробел, предлагая декларацию `\numberwithin` (раздел 8.9.1).

Наконец, стоит отметить, что со счётчиками можно проводить привычные всем математические операции сложения, вычитания, умножения и деления, если загрузить пакет calc, который описан в разделе 7.5.

2.10. Длина

В полиграфии основными единицами длины являются *дюйм* и *пункт*. Дюйм в буквальном переводе с голландского языка означает большой палец. Пункт — единица длины, равная 1/72 части дюйма. В Российской Федерации и ряде европейских стран, где за основу принят французский дюйм (27,1 мм), 1 пункт

приблизительно равен 0,376 мм. В англо-американской системе мер, пользующейся английским дюймом (25,4 мм), 1 пункт приблизительно равен 0,351 мм.

LaTeX построен на англо-американской системе мер. Он использует следующие единицы измерения длины:

`mm`	миллиметр
`cm`	сантиметр
`in`	дюйм (1 in = 2,54 cm)
`pt`	пункт (1 in = 72,27 pt)
`bp`	«большой» пункт (1 in = 72 bp)
`pc`	пайка (1 pc = 12 pt)

Здесь все соотношения между единицами длины даны в виде точных равенств.

Явная длина записывается в виде десятичного числа, за которым следует обозначение единицы измерения. Десятичное число может иметь знак и состоять из любого числа цифр (нужно оставить десятичную точку, если цифр нет вообще). Между знаком (если он есть), числом и обозначением единицы измерения допускаются пробелы, однако пробелы нельзя оставлять между цифрами в числе и между буквами в обозначении единиц измерения. Следующий пример даёт пять образцов правильной записи длины:

```
3 in      29pc       -.667cm      0.mm      + 42.1 pt
```

Нельзя записывать нулевую длину одним нулем без указания единиц измерения — нужно писать `0mm` или `0in` и т. д.

Объекты, измеренные в указанных выше единицах, имеют фиксированную длину, которая не зависит от контекста, в котором применена та или иная команда. Линия длиной в 5 cm при любой модификации исходного текста всегда в печатном документе будет иметь этот размер.

LaTeX знает ещё две единицы измерения, которые скорее являются относительными, чем абсолютными:

`em`	«M-ширина» текущего шрифта
`ex`	«x-высота» текущего шрифта

Каждый шрифт определяет свои собственные значения `em` и `ex`. Достаточно давно величина `em` отождествлялась с шириной буквы «M» текущего шрифта. В настоящее время так считать не совсем верно, хотя приближённо можно попрежнему рассматривать `em` и `ex` как ширину буквы «M» и высоту буквы «x». В семействе шрифтов Computer Modern выдерживается правило, что ширина цифр от 0 до 9 равна 0,5 em, а высота строчной буквы «x» равна 1,0 ex; но для других семейств шрифтов это правило не является абсолютным. Для основного шрифта, которым набрана данная книга, 1 em = 10 pt, 1 ex $\approx$ 4,3 pt, для полужирного шрифта того же размера 1 em = 11,5 pt, 1 ex $\approx$ 4,44 pt, а машинописный шрифт имеет 1 em = 10,5 pt, 1 ex $\approx$ 4,3 pt.

Командная длина суть команда, значение которой есть длина. Командную длину, как и явную, можно использовать в аргументах команд. Ширина отступа

в начале абзаца равна `\parindent`, а команда `1.5\parindent` означает в 1,5 раза бо́льшую длину. Командные длины, как и явные, могут иметь отрицательные значения.

Многие параметры, определяющие стиль документа, такие как ширина отступа в начале абзаца, ширина и высота страницы, расстояние между строками, являются командными длинами. Обычно такие команды задают *нерастяжимую длину*. Случаи, когда команда задаёт *растяжимую длину*, всякий раз будут оговариваться особо. Растяжимая длина имеет *естественную длину* и *степень растяжимости*. Можно представить, что растяжимая длина — это пружинка, у которой есть естественная длина, когда пружинка не деформирована, и есть коэффициент упругости, определяющий, насколько удлинится пружинка, когда её растягивают с определённым усилием.

Величину растяжимой длины записывают с помощью слов

```
plus    minus
```

Слово `plus` означает, что длина может растягиваться, а слово `minus` — сжиматься. Например, выражение `1.5em plus 1pt minus 2pt` означает растяжимую длину, которая в нормальном состоянии имеет размер 1,5 em, но может растягиваться на один пункт и сжиматься на два пункта. Длина `1.5em minus 2pt` может только сжиматься. При формировании страницы LaTeX растягивает или сжимает вертикальные пробелы между абзацами, чтобы все страницы имели одинаковую высоту. Если необходимо растянуть текст на 3 pt, а на страницу попали два пробела с растяжимостью `2pt` и `4pt`, они будут увеличены соответственно на 1 pt и 2 pt, т. е. каждый пропорционально степени своей растяжимости.

Для работы с растяжимыми длинами вполне достаточно двух команд:

```
\fill
\stretch{dec-num}
```

Первая из них есть «бесконечно» растяжимая длина. Она имеет нулевую естественную длину и очень большую степень растяжимости. Примеры использования команды `\fill` даны в разделах 4.3 и 4.6. Команда `\stretch` есть длина с заданной степенью растяжимости. Она имеет нулевую естественную длину и степень растяжимости, составляющую долю *dec-num* от степени растяжимости `\fill`; *dec-num* может быть десятичным числом со знаком. Например, *dec-num* $= -0.5$ соответствует отрицательной длине, имеющей половину «силы» команды `\fill`. Заметим, что простое умножение `\fill` даже на единицу превращает растяжимую длину в нерастяжимую, поэтому `1\fill` есть нулевая длина.

LaTeX располагает тремя декларациями для создания новых и изменения существующих командных длин:

⚠
```
\newlength{len-cmd}
\setlength{len-cmd}{len}
\addtolength{len-cmd}{len}
```

Первая из них объявляет новую командную длину с нулевой начальной длиной. Важно, чтобы команда *len-cmd* не была ранее определена. Например, `\newlength {\abc}` определяет новую командную длину `\abc` с исходной начальной длиной 0. Значение `\abc` может быть изменено затем в любое время при помощи двух других деклараций.

Декларация `\setlength` присваивает *len-cmd* значение *len*. Например, после `\setlength{\abc}{1.2cm}` длина `\abc` принимает значение 1,2 cm.

Декларация `\addtolength` увеличивает значение *len-cmd* на *len*. Если продолжить упражнение с длиной `\abc`, то после `\addtolength{\abc}{-0.5\abc}` она будет равна 0,6 cm.

Существуют ещё несколько команд, которые позволяют измерять ширину и высоту текста, рисунка или другого объекта, по терминологии LaTeX'а являющегося *боксом*. Они описаны в разделе 9.1.1. Если загрузить пакет calc (раздел 7.5), то можно манипулировать с длинами при помощи привычных математических операций сложения, вычитания, умножения и деления, используя при этом обычные обозначения `+`, `-`, `*` и `/`.

А пачпорт у тебя есть?
А. Куприн. Белый пудель

Глава 3

Печатный документ

В этой главе мы рассмотрим общую структуру печатного документа. Напомним, что печатным документом, по терминологии LaTeX'а, называется результат компиляции входного файла. Входной файл содержит исходный текст печатного документа, размеченный командами LaTeX'а. Термины *входной файл* и *исходный текст* часто можно рассматривать как синонимы, но в контексте данной главы их лучше трактовать более чётко. Исходный текст может содержаться в нескольких файлах, а главный, или *корневой*, входной файл помимо исходного текста содержит ещё и преамбулу. На нескольких следующих страницах мы расскажем, как выбрать класс печатного документа, как разбить исходный текст на разделы и какие средства имеет LaTeX, чтобы ускорить работу над большим печатным документом.

3.1. Преамбула

Преамбула начинается с декларации

`\documentclass[options]{class}[release-date]`

которая выбирает класс печатного документа. Она может иметь от одного до трёх аргументов. Обязательный аргумент class задаёт *класс* документа: декларация `\documentclass` указывает, что компилятор должен прочитать файл class`.cls`, который содержит определение тех команд, которые специфичны для каждого класса. Например, в классах даны определения всех команд секционирования. Однако главное содержание файлов class`.cls` составляет определение множества размеров (параметров настройки), начиная от размеров букв и кончая размерами страниц. Файлы с расширением `cls` могут находиться в одном из служебных каталогов LaTeX'а или в текущем (рабочем) каталоге.

Необязательный аргумент options, если он присутствует, модифицирует некоторые параметры настройки печатного документа, принимаемые по умолчанию. Необязательный аргумент может содержать несколько параметров, перечисленных через запятую, то есть в общем случае options есть option_1, option_2... option_N. Список стандартных классов и их опций приведён в разделе 3.2.

Второй необязательный аргумент release-date используется для проверки версии выбранного класса; release-date — это дата выпуска наиболее старой версии

класса, которую можно использовать для компиляции данного входного файла. Дата должна быть записана в формате *гггг*/*мм*/*дд*, где *гггг*, *мм*, *дд* — год, месяц и день выпуска соответственно. Приведём пример декларации `\documentclass` со всеми тремя аргументами:

```
\documentclass[a4paper]{book}[2001/04/21]
```

Вслед за `\documentclass` преамбула обычно содержит одну или несколько деклараций

`\usepackage[`*options*`]{`*package*`}[`*release-date*`]`

Эта декларация действует аналогично `\documentclass`, но только загружает пакет *package*, а именно считывает файл *package*`.sty`. Например, чтобы подготовить статью на русском языке, рекомендуется загрузить пакеты babel и indentfirst, причём первый с опцией `russian`:

```
\documentclass[a4paper]{article}
\usepackage[russian]{babel}
\usepackage{indentfirst}
```

Мы обсудим выбор опций при загрузке классов и пакетов позднее, в соответствующих главах книги,— пока же мы просто демонстрируем, как правильно записать несколько самых первых строчек входного файла.

Пакеты могут иметь общие опции. Например, в дополнение к babel можно загрузить пакет varioref, записав:

```
\documentclass[a4paper]{article}
\usepackage[russian]{babel}
\usepackage[russian]{varioref}
\usepackage{indentfirst}
```

Эту запись можно сократить, так как одна декларация `\usepackage` способна загрузить несколько пакетов:

```
\documentclass[a4paper]{article}
\usepackage[russian]{babel,varioref}
\usepackage{indentfirst}
```

Наконец, пакеты также видят и используют опции, перечисленные непосредственно в `\documentclass` (если, конечно, они знают, что с ними делать), так что можно записать:

```
\documentclass[a4paper,russian]{article}
\usepackage{babel,varioref,indentfirst}
```

Если какая-нибудь из опций не была использована ни классом, ни одним из пакетов, компилятор выдаст предупреждение об этом:

```
LaTeX Warning: Unused global option(s) option¹.
```

Заметим, что вариант

```
\documentclass[a4paper]{article}
\usepackage[russian]{babel,varioref,indentfirst}      (не верно)
```

не годится, так как пакет indentfirst не знает, как обращаться с опцией `russian`. Компилятор выдаст сообщение об ошибке и прекратит обработку документа:

```
! LaTeX Error: Unknown option ‘russian’ for package ‘indentfirst’².
?
```

Порядок загрузки пакетов обычно не играет роли. Однако из любого правила бывают исключения. Например, до недавних пор пакет amsmath (глава 8) следовало загружать до пакета babel с опцией `russian`, иначе возникала ошибка при выборе некоторых шрифтов. В последних выпусках пакета amsmath эта ошибка устранена.

Каждый пакет загружается только раз. Если один и тот же пакет запрошен повторно, ничего не произойдёт (повторная загрузка не будет выполнена) при условии, что пакет не затребован с опцией, отличной от использованной в первый раз. Если такая опция заказана, то LaTeX выдаст сообщение об ошибке:

```
! LaTeX Error: Option clash for package package³.
?
```

Поскольку пакеты могут загружаться как из входного файла, так и из класса или из других пакетов, причину этой ошибки бывает нелегко распознать. Введя `h` в ответ на `?` в сообщении об ошибке, можно получить исчерпывающую информацию о том, какие опции запрошены. Попытаться исправить ситуацию можно, загрузив с самого начала пакет со всеми нужными необязательными аргументами.

Преамбула заканчивается командой `\begin{document}`. Процедура

`\begin{document} ... \end{document}`

форматирует текст печатного документа. Помимо `\usepackage` преамбула может содержать другие декларации (например, определения новых команд). Некоторые декларации могут располагаться только в преамбуле. С примерами таких деклараций Читатель познакомится в конце этой главы.

[1] Неиспользованная глобальная опция *option*.
[2] Неизвестная опция russian для пакета indentfirst.
[3] Конфликт опций для пакета *package*.

3.2. Стандартные классы

Стандартными являются 6 классов:

`article`	(статья)	`book`	(книга)
`report`	(отчёт)	`letter`	(письмо)
`proc`	(доклад)	`slides`	(слайды)

Специфическим особенностям каждого из них посвящена глава 15, а предшествующие ей главы описывают способы форматирования печатного документа, доступные при выборе практически любого класса.

Стандартные классы могут иметь опции, перечисленные ниже. Альтернативные опции, из которых только одна может быть указана в аргументе *options* команды `\documentclass`, разделены вертикальной чертой.

`10pt` | `11pt` | `12pt` устанавливает, что базовым (т. е. `\normalsize`) является шрифт размером 10pt, 11pt или 12pt. По умолчанию используется `10pt`. (Эти опции не распознаются классом `slides`.)

`letterpaper` | `legalpaper` | `executivepaper` | `a4paper` | `a5paper` | `b5paper` устанавливает размер листа бумаги, для которого будет сформатирован текст, в соответствии со следующей таблицей:

Letter	8,5 in×11 in	A4	210 mm×297 mm
Legal	8,5 in×14 in	A5	148 mm×210 mm
Executive	7,25 in×10,5 in	B5	176 mm×250 mm

По умолчанию используется `letterpaper`. В нашей стране размер бумаги `letterpaper` почти не встречается, поэтому почти всегда следует указывать опцию `a4paper`. Опции `a5paper` и `b5paper` не поддерживаются классом `proc`.

`landscape` переставляет значения ширины и высоты листа бумаги, задавая так называемую альбомную ориентацию страницы. При этом текст будет сформирован для листа бумаги выбранного размера, повёрнутого на 90°.

`final` | `draft` определяет режим маркировки проблемных для форматирования строк документа. Если LaTeX не может найти хорошее место, чтобы разорвать строку, он формирует строку, которая простирается за правую границу колонки текста. Предупреждение об этом выводится на экран. При использовании опции `draft` строки, вызвавшие затруднения с переносом, будут помечены в печатном документе закрашенным прямоугольником у края строки. Опция `final`, которая не метит такие строки, используется по умолчанию.

`oneside` | `twoside` форматирует документ для печати на одной или обеих сторонах листа. Во втором случае, когда используется опция `twoside`, чётные и нечётные страницы по-разному размещаются на листах бумаги, чтобы

облегчить их брошюровку. По умолчанию используется `oneside` за исключением класса `book`, где по умолчанию предусмотрена двусторонняя печать. Опция `twoside` не может использоваться с классом `slides`.

`openright` | `openany` указывает, что главы должны начинаться на правой странице (`openright`) или на любой (`openany`). Эти опции применяются только с классами `report` (где по умолчанию действует `openany`) и `book` (где действует `openright`), так как главы (начинающиеся с команды секционирования `\chapter`) имеются только в документах этих двух классов.

`onecolumn` | `twocolumn` печатает документ в одну (`onecolumn`) или две (`twocolumn`) колонки (раздел 17.3). Все классы, кроме `proc`, по умолчанию настроены на печать в одну колонку. Опция `twocolumn` не поддерживается классами `slides` и `letter`, а опция `onecolumn` не поддерживается классом `proc`.

`titlepage` | `notitlepage` изменяет действие команды `\maketitle` и процедуры `abstract`, которые печатают, соответственно, заголовок и аннотацию к документу. Если действует опция `titlepage`, заголовок и аннотация размещаются на отдельных страницах, в противном случае — непосредственно перед основным текстом. По умолчанию `titlepage` используется для всех классов, кроме `article`. Опция `titlepage` не поддерживается классом `proc`. Обе опции не поддерживаются классом `letter`, где не определена команда `\maketitle`.

`openbib` форматирует список литературы в так называемом открытом стиле (глава 13), когда элементы списка форматируются в виде нескольких блоков, каждый из которых начинается с новой строки. Эта опция не распознается классами `slides` и `letter`.

`leqno` устанавливает, что номера формул в математических уравнениях печатаются слева, а не справа (глава 6).

`fleqn` выравнивает формулы по левому краю, а не по центру (глава 6).

3.3. Пакеты

На CTAN'е можно найти более 1 000 пакетов. Около двухсот из них описано на шестистах страницах книги «Путеводитель по пакету LaTeX и его расширению LaTeX 2$_\varepsilon$» [12]. Здесь мы перечислим пакеты, входящие в минимальный комплект, распространяемый в составе любой реализации системы LaTeX, дадим их краткую характеристику и указание, в какой главе имеется более подробное описание[4]. Помимо этих, так называемых стандартных пакетов, в последующих главах мы расскажем ещё о нескольких, которые, как нам кажется, могут быть полезны

[4] Наиболее детальную документацию по всем пакетам можно получить, выполнив компиляцию файлов с исходным кодом пакетов. Они имеют расширение `dtx`.

широкому кругу пользователей. Полный список всех пакетов, которые упоминаются в нашей книге, можно найти в алфавитном указателе под словом «пакеты».

alltt — Этот пакет определяет процедуру `alltt`, аналогичную `verbatim` за тем исключением, что обратный слеш \, фигурные скобки `{` и `}` имеют своё обычное значение (раздел 5.6.2).

doc — Используется для печати документации по LaTeX'у. Описан в файле `doc.dtx` и в книге [12].

fixltx2e — Вносит некоторые изменения в способ оформления документа, принятый в настоящее время форматом LaTeX. По большей части такие изменения относятся к документам, сформатированным в две колонки (раздел 17.3).

flafter — Гарантирует, что любой плавающий объект, т. е. рисунок или таблица, оформленные с помощью процедур `figure` или `table`, не появится в печатном документе до первой ссылки на него (раздел 11.1).

fontenc — Используется для выбора внутренней кодировки шрифтов LaTeX'а (раздел 16.5.1).

graphpap — Определяет команду `\graphpaper`, которая печатает масштабную сетку и может использоваться в процедуре `picture` (раздел 9.5.3).

ifthen — Вводит команды типа «if... then... else...», результат которых зависит от выполнения программируемого условия (раздел 7.4).

inputenc — Используется для указания кодировки исходного текста во входном файле (раздел 16.5.2).

latexsym — Вводит команды для печати некоторых редких символов, имевшиеся в предыдущей версии LaTeX 2.09, но отсутствующие в формате LaTeX 2_ε (глава 6).

makeidx — Используется для генерации алфавитного указателя (глава 14).

newlfont — Используется для эмуляции команд переключения шрифтов в предыдущей версии LaTeX 2.09 с «ортогональной схемой выбора шрифтов» (приложение A).

oldlfont — Используется для эмуляции команд переключения шрифтов в LaTeX 2.09 (приложение A).

shortvrb — Вводит краткий вариант команды `\verb` (раздел 5.6.4).

showidx — Печатает аргумент каждой команды `\index`, которая метит термины для алфавитного указателя, на полях той страницы, куда эта команда попадает (раздел 14.3).

syntonly — Используется для обработки входного файла с целью его проверки без генерации печатного документа (приложение B.2).

tracefnt — Регулирует объём информации о загружаемых шрифтах, выводимой на экран при компиляции исходного текста (раздел 16.7).

Вместе с перечисленными пакетами распространяются ещё несколько коллекций пакетов: AMS-LaTeX, AMSFonts, babel, cyrillic, graphics, PSNFSS и tools. Они входят в минимальный комплект поставки системы LaTeX.

3.3.1. Коллекция пакетов AMS-LaTeX и AMSFonts

Пакеты, разработанные Американским математическим обществом, используются для печати сложных математических текстов. Основным в коллекции amslatex является пакет amsmath. Он вводит несколько новых процедур для форматирования математических формул и множество новых команд для печати математических символов. Пакеты коллекции AMSFonts загружают шрифты для печати готических и других «экзотических» символов. Все имеющиеся символы определены в пакете amssymb. Пакеты Американского математического общества описаны в главе 8.

3.3.2. Коллекция пакетов babel

Коллекция babel содержит пакеты поддержки многих языков мира, позволяя в одном документе использовать несколько языков. Коллекция содержит по одному пакету для каждого поддерживаемого языка, например: english (английский язык), greek (греческий), russianb (русский). Однако загружать следует головной пакет babel, указывая нужные языки в необязательном аргументе команды `\usepackage`. Пакет babel описан в разделе 3.6.

3.3.3. Коллекция пакетов cyrillic

Начиная с декабрьского выпуска 1998 года LaTeX включает коллекцию пакетов cyrillic, призванную обеспечить стандартизацию способа русификации системы LaTeX. Строго говоря, cyrillic не является коллекцией пакетов, не имея в своём составе ни одного пакета. Коллекция содержит определения (файлы с расширением `def`) всех мыслимых кодировок, которые встречаются в исходных текстах на славянских языках. Кодировку исходного текста нужно указывать в виде опции при загрузке пакета inputenc. Кодировки, добавленные коллекцией cyrillic, перечислены в разделе 16.5.2, а проблемы русификации в концентрированном виде обсуждаются в главе 4.

3.3.4. Коллекция пакетов graphics

Коллекция включает три основных пакета: graphics, graphicx и color. Два первых являются альтернативными, обладая одинаковыми функциями, но используя разный интерфейс (метод взаимодействия) с пользователем. Они обеспечивают поддержку включения, масштабирования и вращения графических изображений, созданных другими графическими программами. Пакет color предназначен для работы с цветными текстами. Графические пакеты описаны в главе 10.

3.3.5. Коллекция пакетов psnfss

Коллекция предоставляет всё необходимое для печати текстов с использованием разнообразных PostScript шрифтов (кроме самих шрифтов). Описана в разделе 16.8.

3.3.6. Коллекция пакетов tools

Коллекция tools содержит ряд пакетов, расширяющих возможности некоторых процедур и команд LaTeX'а.

afterpage	Размещает текст в начале следующей страницы (раздел 11.3.1).
array	Расширенная версия процедур `array`, `tabular` и `tabular*` с множеством дополнительных функций (раздел 12.3).
bm	Вводит команду `\bm` для набора полужирных математических символов (раздел 6.6).
calc	Полезен для программирования особо сложных команд (раздел 7.5).
dcolumn	Упрощает выравнивание колонок в таблицах по десятичной точке в числах (раздел 12.3.2).
delarray	Добавляет большие скобки вокруг матриц (раздел 12.3.5).
enumerate	Расширенная версия процедуры `enumerate` (раздел 5.6.1).
fontsmpl	Используется разработчиками шрифтов для их проверки (описан в `fontsmpl.dtx`).
ftnright	Размещает все подстрочные примечания в правой колонке при двухколоночной печати (раздел 4.8).
hhline	Позволяет разнообразить разделительные линии в таблицах и варианты их пересечений (раздел 12.3.4).
indentfirst	Вводит отступ в начале первого абзаца каждого раздела. По умолчанию LaTeX не делает отступ в начале первого абзаца (раздел 4.5).
layout	Печатает диаграмму, иллюстрирующую все параметры стиля страницы, установленные выбранным классом печатного документа (раздел 17.2).
longtable	Формирует многостраничные таблицы. Использует расширенные возможности пакета `array`, если тот также загружен (раздел 12.5).
multicol	Печатает текст в заданном числе колонок сбалансированной длины (раздел 17.4).
rawfonts	Производит загрузку шрифтов, используя метод, принятый в предыдущей версии LaTeX 2.09 (приложение A).
showkeys	Печатает «ключи», используемые командами `\label`, `\ref`, `\cite` и т. д.; полезен при работе с черновой копией печатного документа (раздел 3.7).

tabularx	Вводит процедуру tabularx для создания таблиц заданной ширины (раздел 12.4).
theorem	Расширенный вариант `\newenvironment` для определения новых процедур (раздел 7.3.1).
trace	Используется разработчиками пакетов для их отладки (описан в файле `trace.dtx`).
varioref	Вводит дополнительные команды для создания ссылок на номера страниц (раздел 3.7.1).
verbatim	Делает возможным применение процедуры `verbatim` к большим массивам текста (раздел 5.6.3).
xr	Организует перекрёстные ссылки между несколькими печатными документами (раздел 3.7.2).
xspace	Позволяет контролировать удаление или добавление пробелов после имени команды (раздел 7.1).

3.3.7. Пакеты contributed

Перечисленные пакеты относятся к так называемым стандартным пакетам. Они должны присутствовать в минимальном варианте установки системы LaTeX.

Помимо стандартных существуют сотни дополнительных пакетов. Их можно найти на серверах CTAN, перечисленных на стр. 8. Невозможно описать и даже просто перечислить все такие пакеты. Но все-таки некоторым мы решили уделить внимание в нашей книге, поскольку они особенно часто востребованы нашими коллегами.

Некоторые реализации системы LaTeX автоматизируют процесс установки дополнительных пакетов, имея в своём дистрибутиве практически все существующие пакеты. Например, MiKTeX содержит мастер установки пакетов MiKTeX Package Manager, который позволяет находить нужные пакеты по названиям, ключевым словам или иным признакам. Но мы расскажем, как установить пакет «своими руками».

На серверах CTAN дополнительные пакеты размещены в каталоге `\tex\macros\latex\contrib`, где размещены два каталога: `supported` и `other`. В первом находятся так называемые поддерживаемые пакеты; во втором — по большей части устаревшие, которые не рекомендуется использовать без крайней необходимости.

На компьютере пользователя пакеты размещаются в каталоге `\tex\latex` так называемого корневого каталога `texmf`. Одновременно может существовать несколько корневых каталогов. Например, мастер установки MiKTeX рекомендует создать дополнительный корневой каталог `localtexmf`. Именно туда целесообразно устанавливать пакеты, шрифты и т. п. «вручную». Тогда при обновлении версии MiKTeX достаточно заменить только первый корневой каталог `texmf` (с нижележащими каталогами), не затрагивая `localtexmf`.

Для установки нового пакета его нужно скопировать в подкаталог `\tex\latex` корневого каталога `localtexmf` (или `texmf`). При этом целесообразно сохранить структуру каталогов. Например, каталог `\tex\macros\latex\contrib\supported\cite` с сервера CTAN нужно целиком скопировать в каталог `\localtexmf\latex\contrib\supported\cite` на компьютере пользователя. Так труднее запутаться при обновлении пакетов. Затем необходимо зарегистрировать добавленные файлы, что в большинстве современных реализаций системы LaTeX производится путём обновления базы данных. В системе MiKTeX эта операция производится из меню Пуск | Программы | MiKTeX | MiKTeX Options | Refresh Now. После регистрации пакета его можно загружать для компиляции любого документа обычным способом при помощи `\usepackage`.

3.4. Титульная страница и аннотация

Печатный документ обычно начинается с заголовка. Стандартный заголовок печатает команда

`\maketitle`

Заголовок размещается на отдельной титульной странице, если действует опция `titlepage`. Напомним, что эта опция выбирается по умолчанию всеми стандартными классами, кроме `article` и `proc`. Если выбрана опция `notitlepage`, команда `\maketitle` начинает новую страницу, присваивает ей номер 1 и печатает заголовок в её верхней части. Команда `\maketitle` обычно стоит сразу же после `\begin{document}`. Информацию для печати заголовка она получает из нескольких команд, которые должны предшествовать ей:

`\title{`*title*`}` объявляет название печатного документа. В аргументе *title* допускается использование команды `\\`, чтобы указать компилятору, как разбить длинное название на строки. Название может быть пустым (когда в фигурных скобках ничего нет), но сама команда обязательна, если команда `\maketitle` будет использована.

`\author{`*author*`}` объявляет автора или список авторов. Команду `\\` применяют для разбиения списка авторов на строки, а также для того, чтобы в отдельной строке напечатать их адреса или другую информацию. Команда `\author` обязательна при использовании команды `\maketitle`, но её аргумент может быть пустым.

`\date{`*date*`}` объявляет аргумент *date* датой выпуска печатного документа. Если данная команда отсутствует, LaTeX проставит в заголовке текущую дату. Если дата не нужна, аргумент *date* должен быть пустым. Некоторые нестандартные классы документов не печатают дату, даже если использована команда `\date` с непустым аргументом.

Команды `\title`, `\author`, `\date` сами ничего не печатают, поэтому их, казалось бы, можно поместить в преамбулу. Однако некоторые нестандартные классы,

```
\title{\LaTeX\thanks{Версия  2.09.}
  \\ для всех}
\author{К.\,О.~Тельников
  \and П.\,З.~Чеботаев
 \\ \itshape Новосибирск}
\date{1994}
\maketitle
```

LaTeX*

для всех

К. О. Тельников   П. З. Чеботаев

Новосибирск

1994

* Версия 2.09.

Рис. 3.1. Пример титульной страницы

в том числе класс `revtex4`, описанный в разделе 15.5, не допускают подобную вольность. Аргументы этих и только этих трёх команд могут содержать команду

`\thanks{`*text*`}`, которая печатает подстрочное примечание к заголовку. Её можно использовать для указания адресов авторов, благодарностей за финансовую или иную поддержку и пр. Аргумент команды `\author` может также содержать команду

`\and` между фамилиями авторов, которые в этом случае печатаются в виде таблицы.

Пример заголовка с использованием всех возможностей приведён на рисунке 3.1.

Чтобы получить полный контроль над форматом и содержанием титульной страницы, лучше всего использовать процедуру `titlepage`:

`\begin{titlepage}  ...  \end{titlepage}`

Она подготавливает совершенно чистую страницу, присваивает ей номер 1, но не печатает его, так как вид титульной страницы полностью определяется содержанием тела процедуры `titlepage`.

В небольших печатных документах, таких как статьи в журналах, вслед за заголовком часто следует аннотация. Аннотацию печатает процедура `abstract`:

`\begin{abstract}  ...  \end{abstract}`

Однако она не определена в классах `book` (книга) и `letter` (письмо), где она не нужна. Действительно, основному содержанию книги обычно предшествует предисловие, а не аннотация. Об особенностях оформления предисловия мы расскажем в разделе 3.10.

Аннотация помещается на отдельной странице в печатном документе, если действует опция `titlepage`. Если же действует опция `notitlepage`, процедура `abstract` сама по себе не начинает новую страницу. Поэтому, если она стоит перед командой `\maketitle`, текст аннотации окажется на отдельной странице, так как `\maketitle` печатает заголовок всегда с новой страницы. Если же процедура

стоит после `\maketitle`, то аннотация будет напечатана сразу после заголовка. Нестандартные классы могут изменять описанный порядок. Например, в классе `revtex4` процедура `abstract` должна быть размещена до команды `\maketitle`, так как именно она печатает текст, взятый из тела процедуры `abstract`.

Команды, описанные в данном разделе, не определены в классе `letter`.

3.5. Команды секционирования

Стандартные классы поддерживают следующие команды секционирования, перечисленные в порядке старшинства:

⚠	\part	[toc]{*head*}	\part*	{*head*}
⚠	\chapter	[toc]{*head*}	\chapter*	{*head*}
⚠	\section	[toc]{*head*}	\section*	{*head*}
⚠	\subsection	[toc]{*head*}	\subsection*	{*head*}
⚠	\subsubsection	[toc]{*head*}	\subsubsection*	{*head*}
⚠	\paragraph	[toc]{*head*}	\paragraph*	{*head*}
⚠	\subparagraph	[toc]{*head*}	\subparagraph*	{*head*}

Каждая из команд в левой колонке начинает новый раздел, то есть присваивает разделу очередной номер, печатает его заголовок *head*, заносит этот заголовок в колонтитулы (раздел 17.1) и в оглавление (раздел 3.10). Если имеется необязательный аргумент toc, то именно он заносится в оглавление и колонтитулы. Например, одна из подсекций в главе 1 называется «Глава, секция и др.», а в оглавлении подсекция с тем же номером 1.12.2 называется «Секционирование текста». Таков результат действия команды

```
\subsection[Секционирование текста]{Глава, секция и др.}
```

Форма команды секционирования со звездочкой `*` только печатает заголовок, ничего не записывая в оглавление и в колонтитулы и не увеличивая значение счётчика, нумерующего разделы соответствующего уровня.

Формат номера и заголовка раздела определяется классом документа. Обычно нумерация устроена иерархически, то есть перед номером младшего раздела стоят номера старших разделов, отделённые друг от друга точкой. Когда исполняется одна из команд секционирования, соответствующий этой команде счётчик увеличивается на единицу, а счётчики младших разделов обнуляются. В этом смысле раздел младшего уровня находится внутри старшего. Метод нумерации разделов, как и вообще любых нумерованных объектов, обсуждался в разделе 2.9. Там же рассказано, как можно изменить формат печати номеров разделов. Счётчик `secnumdepth`, описанный в разделе 3.10, определяет младший из нумеруемых разделов. Раздел `\part` (часть) необязателен и, формально являясь самым старшим, не влияет на нумерацию младших разделов: если последней в Части 1 была Глава 4 (`\chapter`), то Часть 2 начнётся с Главы 5. Поэтому самой старшей обязательной командой секционирования является `\chapter`. Однако в

классах `article` (статья) и `proc` (доклад) она не определена, поскольку статьи и доклады не делят на главы, а самой старшей является команда `\section` (секция). Зато статьи удобно включать в книги в виде глав, заменив `\maketitle` на `\chapter`. В классе `letter` команды секционирования вообще не определены.

Необязательный аргумент команд секционирования является подвижным, так как он переписывается в оглавление и колонтитулы. Если необязательный аргумент отсутствует, то подвижным становится обязательный аргумент. Хрупкие команды (раздел 2.7) не работают в подвижном аргументе других команд, если их не защитить командой `\protect`. Необязательный аргумент к команде секционирования обычно добавляют в тех случаях, когда пытаются достичь каких-то особых визуальных эффектов в заголовках разделов или если заголовок слишком длинный, чтобы уместиться в одной строке. В заголовках нелепо смотрятся знаки переноса, поэтому в длинном заголовке желательно явно указать, где должен быть переход на новую строку, используя команду `\\`, а для оглавления и колонтитула заготовить укороченный вариант. Кроме всего прочего, места переноса в самом заголовке и в ссылке на него в оглавлении вовсе не обязаны совпадать. Рецепт решения всех подобных проблем даёт следующий пример:

```
\section[Это укороченный заголовок]
{Это очень, очень, очень, очень\\ очень, очень длинный заголовок}
```

Если в печатном документе имеется приложение, то оно начинается с декларации

```
\appendix
```

и использует те же команды секционирования, что и основная часть текста. Декларация `\appendix` изменяет способ нумерации разделов. После неё начинается новый отсчёт разделов, причём первая секция обозначается буквой A, вторая — B, третья — C и т. д. Пакет babel с опцией `russian` заменяет слово «Appendix» перед номером главы в приложении на «Приложение».

Ключевые слова, такие как «Глава» или «Приложение», которые LaTeX автоматически печатает в заголовках разделов, хранятся в командах-логосах. Имя соответствующей команды-логоса обычно получается присоединением слова `name` к имени команды, использующей ключевое слово. В стандартных классах определены логосы

```
\abstractname
\partname
\chaptername
\appendixname
```

Они могут быть изменены при помощи команды `\renewcommand` (раздел 7.1). Например, если требуется, чтобы перед названием глав печаталось слово «Часть», нужно действовать по следующему образцу:

```
\renewcommand{\chaptername}{Часть}
\verb|\chapter| использует
ключевое слово \chaptername.
```

Команда `\chapter` использует ключевое слово Часть.

Можно также изменить способ нумерации глав в приложении и заменить латинские буквы русскими. Читателю, знакомому с содержанием предыдущей главы (раздел 2.9), уже известно, что для этого нужно переопределить команду `\thechapter`:

```
Глава~\thechapter.
\renewcommand{\thechapter}{\Asbuk{chapter}}
стала главой~\thechapter.
```

Глава 3 стала главой В.

Раз уж мы в очередной раз вспомнили о пакете babel, рассмотрим его повнимательнее.

3.6. «Вавилонское столпотворение»

Именно так переводится на русский язык название пакета babel, предназначенного для подготовки многоязычных документов LaTeX. Пакет был создан Йоханнесом Брамсом (Braams, Johannes) при поддержке разработчиков из многих стран мира. В разработке его русской части принимали участие Ольга Лапко, Владимир Волович и Вернер Лемберг (Lemberg, Werner).

Любой язык, который используется в документе, должен быть указан в необязательном аргументе декларации `\usepackage`, которая загружает пакет babel. Например, в преамбулу документа на англо-русском гибриде языков необходимо вставить команду `\usepackage[english,russian]{babel}`:

```
\documentclass{article}
. . .
\usepackage[english,russian]{babel}
. . .
\begin{document}
```

Нужные языки можно также указать в необязательном аргументе декларации `\documentclass`:

```
\documentclass[english,russian]{article}
. . .
\usepackage{babel}
. . .
\begin{document}
```

В начале документа будут действовать установки для языка, указанного последним, поэтому порядок перечисления языков имеет значение. В нашем примере последним назван русский язык. Соответственно, вначале будет активизирована таблица переносов русского языка, а ключевые слова, типа упоминавшегося выше `\appendixname`, будут русскими.

На момент публикации нашей книги пакет babel поддерживал 60 языков и диалектов:

```
acadian afrikaans american austrian
bahasa basque brazil brazilian breton british bulgarian
canadian canadien catalan croatian czech
danish dutch english esperanto estonian
finnish francais frenchb french
galician german germanb greek polutonikogreek
hebrew hungarian icelandic irish italian
latin lowersorbian magyar naustrian ngerman norsk
nynorsk polish portuges portuguese romanian russian
samin scottish serbian slovak slovene spanish swedish
turkish ukrainian uppersorbian welsh UKenglish USenglish
```

(babel)

Пользовательский интерфейс к пакету babel очень прост. Достаточно знать, что команда `\selectlanguage` служит для выбора языка, а `\iflanguage` позволяет напечатать тот или иной вариант текста в зависимости от того, какой язык является текущим, т. е. выбран в данный момент. Впрочем, статус нашей книги как полного руководства по LaTeX'у обязывает нас рассказать и о других командах, которые вводит пакет babel.

3.6.1. Смена языка

Переключение на заданный язык производит декларация

`\selectlanguage{`*language*`}` (babel)

где параметр *language* может быть названием одного из языков, указанных в преамбуле документа. В нашем примере можно выбирать значения `english` или `russian`, поскольку только они были названы в аргументе `\usepackage`. Область действия выбранного языка подчиняется обычным правилам группирования, т. е. её можно ограничить при помощи фигурных или командных скобок:

```
<<\appendixname>> по-английски будет
{\selectlanguage{english}``\appendixname''}.
<<\appendixname>>\ldots
```

«Приложение» по-английски будет “Appendix”. «Приложение»...

Переключение языка может сопровождаться более глубинными изменениями, нежели простая замена ключевых слов и правил переноса слов по слогам. Мы недаром заменили в предыдущем примере русские кавычки «...» на иностранные “...” после переключения на английский язык, поскольку смена языка может сопровождаться сменой кодировки шрифта (как при переходе от русского языка к английскому). Иначе символы `<` и `>` в исходном тексте отобразились бы в печатном документе в виде знаков ¡ и ¿ (которые в испанском языке ставят, соответственно, перед восклицательным и вопросительным предложением):

```
<<--->> по-иностранному будет
{\selectlanguage{english}<<--->>}.
```

«—» по-иностранному будет ¡¡—¿¿.

Стоит также обратить внимание на то, что при переключении на русский язык длина тире автоматически уменьшается.

При смене языка посредством `\selectlanguage` выполняются следующие операции:

- переключение таблицы переносов;
- замена кодировки шрифтов, если это необходимо;
- добавление (удаление) команд и лигатур, имеющих смысл только для шрифтов с определённой кодировкой или для определённого языка;
- замена ключевых слов, таких как «Глава» или «Содержание», которые компилятор автоматически вставляет в документ;
- переопределение команды `\today`, которая печатает текущую дату;
- смена типографских настроек, учитывающих национальные традиции, таких как длина тире, величина пробелов вокруг тире или после точки в конце предложения.

Поскольку выделение области действия декларации `\selectlanguage` не очень удобно в тех случаях, когда эта область велика, пакет babel предоставляет альтернативный способ переключения языка при помощи процедуры

`\begin{otherlanguage}{`*language*`}   ...   \end{otherlanguage}` (babel)

Для коротких включений на ином языке существует команда

`\foreignlanguage{`*language*`}{`*text*`}` (babel)

Параметр *language* здесь имеет тот же смысл, что и в аргументе `\selectlanguage`, но действует только в теле процедуры `otherlanguage` или втором аргументе *text* команды `\foreignlanguage`.

Процедура `otherlanguage` выполняет все те же действия, что и декларация `\selectlanguage`, а команда `\foreignlanguage` экономит на замене ключевых слов. Так же поступает *-вариант процедуры `otherlanguage`:

`\begin{otherlanguage*}{`*language*`}   ...   \end{otherlanguage*}` (babel)

Процедура

`\begin{hypenrules}{`*language*`}   ...   \end{hypenrules}` (babel)

переключает только правила переноса слов. Её можно использовать для отключения переносов в части текста; для этого в аргументе *language* нужно указать `nohyphenation`.

Если заранее неизвестно, какой язык является текущим, можно использовать команду

`\iflanguage{`*language*`}{`*true-clause*`}{`*false-clause*`}` (babel)

Если текущий язык совпадает с *language*, она печатает текст второго аргумента *true-clause*; в противном случае в ход идёт текст третьего аргумента *false-clause*. Команду `\iflanguage` удобно применять для определения новых команд (глава 7), которые должны быть восприимчивы к выбору текущего языка.

Имя текущего языка хранится в команде

`\languagename` (babel)

что подтверждает следующий пример:

```
\languagename{} \foreignlanguage{english}{\languagename}
```

russian english

3.6.2. Активные сокращения

Пакет babel может вводить ещё ряд команд в зависимости от выбранного языка. Примером могут служить команды для набора кавычек. „Лапки“ можно набрать при помощи команд `\glqq` и `\grqq` соответственно, а «ёлочки» — командами `\flqq` и `\frqq`.

```
\glqq{}Лапки\grqq{} и \flqq{}ёлочки\frqq{}
перекочевали в русскую версию пакета
\textsf{babel} соответственно из немецкой
и французской версий.
```

„Лапки“ и «ёлочки» перекочевали в русскую версию пакета babel соответственно из немецкой и французской версий.

Вероятно, найдется не много приверженцев набора кавычек при помощи команд. Поэтому пакет babel предоставляет альтернативный способ. Для многих языков, в том числе и русского, он вводит так называемые сокращения (shorthands). Сокращение по сути является командой, но начинается, с символа двойных кавычек `"`, а не с обратного слеша, как обычные команды LaTeX'а. С помощью сокращений последний пример можно переписать следующим образом:

```
"`Лапки"' и "<ёлочки">
```

„Лапки“ и «ёлочки»

После сокращений не нужно ставить `{}` или каким-то иным способом обозначать их окончание, что приходится проделывать с обычными командами, которые иначе «съедают» пробелы после себя.

Платой за удобства является то, что символ `"` становится «активным», т. е. ведёт себя в некоторых отношениях подобно обратному слешу `\`. Обычно это никак не проявляется, но в технически изощрённых документах, подобных нашей книге, где приходится докапываться до «сути» LaTeX'а, доставляет массу неудобств. К счастью для «изощренцев» активность символа `"` можно отключать и включать соответственно при помощи деклараций

`\shorthandoff{`*shorthand*`}`
`\shorthandon{`*shorthand*`}` (babel)

что мы сразу и продемонстрируем:

```
\shorthandoff{"} "‘Лапки"’ и "<ёлочки">
```
"‘Лапки"’ и "<ёлочки">

Декларации

```
\useshorthands{s}
\aliasshorthand{s1}{s2}
\defineshorthand{shorthand}{cmd}
```
(babel)

позволяют вводить новые сокращения. Первую из них, в отличие от двух других, следует использовать только в преамбуле документа. Она объявляет, что сокращения могут начинаться с символа s. Например, после `\useshorthands{/}` символ «/» можно использовать для создания новых сокращений. Вторая декларация объявляет, что сокращения, начинающиеся с символа s_1, можно использовать как синонимы существующих сокращений, начинающихся с символа s_2. Например, после `\aliasshorthand{"}{/}` сокращения `/<` и `"<` будут иметь одинаковый эффект:

```
\aliasshorthand{"}{/} /‘Лапки/’ и /<ёлочки/>
```
„Лапки“ и «ёлочки»

Наконец, декларация `\defineshorthand` сопоставляет сокращение *shorthand* команде *cmd*, причём сокращение *shorthand* может состоять из одного или двух символов:

```
\defineshorthand{"1}{\flqq}
\defineshorthand{"2}{\frqq}
Такие вот "1ёлочки"2!
```
Такие вот «ёлочки»!

Декларация

```
\languageshorthands{language}
```
(babel)

задействует все сокращения, заданные для языка *language*; не стоит повторять, что *language* должен быть перечислен в списке языков при загрузке пакета babel. Следующий пример станет понятен, если учесть, что при выборе английского языка сокращений нет:

```
\languageshorthands{english} "<Ёлочек"> нет.
```
"<Ёлочек"> нет.

3.6.3. Таблицы переносов

Мы вскользь упомянули, что переключение языка сопровождается сменой таблицы переносов. Она используется алгоритмом переноса слов по слогам. Все необходимые таблицы переносов должны быть установлены при генерации формата LaTeX, иначе компилятор даже не будет пытаться переносить слова. Хотя поддержка русского языка с 1999 года является обязательной составной частью любой версии LaTeX’а, по умолчанию алгоритм переноса русских слов, как правило, не устанавливаются. Чтобы иметь возможность работать с русскими текстами, после завершения стандартной процедуры установки необходимо предпринять ещё ряд шагов, перечисленных в приложении B.1.

3.6.4. Замена ключевых слов

Мы неоднократно писали (последний раз на странице 75), что для замены ключевых слов, которые LaTeX автоматически вставляет в самых разных местах, необходимо изменить определение соответствующей команды `\`*cmd*`name` при помощи `\renewcommand`. Это истинная правда, но лишь часть всей правды. Действительно, команды `\`*cmd*`name` печатают разный текст в зависимости от текущего языка, поэтому переопределение любой такой команды посредством `\renewcommand` остается в силе лишь до следующего переключения языка. Далее новое значение будет потеряно, поскольку любая из команд выбора языка восстанавливает значения всех команд `\`*cmd*`name`, заданные пакетом babel для соответствующего языка:

```
1. \renewcommand{\appendixname}{Дополнение}\appendixname\\
2. \selectlanguage{english}\appendixname\\
3. \selectlanguage{russian}\appendixname
```

1. Дополнение
2. Appendix
3. Приложение

Список определений всех команд `\`*cmd*`name` для конкретного языка *language* хранится в команде

`\captions`*language* (babel)

В частности, для русского языка — это команда `\captionsrussian`, для английского — `\captionsenglish` и т. д. Имеется также команда

`\date`*language* (babel)

которая содержит определение команды `\today`, печатающей текущую дату на языке *language*, и команда

`\extras`*language* (babel)

содержащая все прочие определения, необходимые для заданного языка, такие как типографские сокращения для набора диакритических знаков.

Добавление или изменение определений, хранящихся в указанных трёх командах, производится при помощи команды

`\addto{`*cmd*`}{`*defs*`}` (babel)

где *cmd* обозначает любую из команд `\captions`*language*, `\date`*language* или `\extras`*language*, а второй аргумент *defs* должен содержать дополнительные определения списка команд, хранимых в `\captions`*language*. Что-либо делать с командами `\date`*language* и `\extras`*language* вряд ли стоит, поэтому мы ограничимся модификацией приведённого выше примера, чтобы показать, как переименовать ключевое слово «Приложение» раз и навсегда:

```
\addto{\captionsrussian}{
    \renewcommand{\appendixname}{Дополнение}}
1. \selectlanguage{russian}\appendixname\\
2. \selectlanguage{english}\appendixname\\
3. \selectlanguage{russian}\appendixname
```

1. Дополнение
2. Appendix
3. Дополнение

Любые изменения, произведённые через \captions*language*, \date*language* или \extras*language*, вступают в силу после первого явного переключения на соответствующий язык. Именно поэтому мы добавили команду \selectlanguage перед первым применением \appendixname.

Можно рекомендовать собрать все изменения ключевых слов в преамбуле входного файла вслед за командой \usepackage, которая загружает пакет babel. Во-первых, там легче отслеживать подобные изменения. Во-вторых, команда \begin{document} автоматически активизирует язык, указанный последним в необязательном аргументе \usepackage, выполняя команду \selectlanguage с соответствующим аргументом. Поэтому любые изменения, произведенные с помощью \addto немедленно вступят в силу. Напротив, изменение ключевых слов посредством команды \renewcommand, размещённой в преамбуле, но вне второго аргумента \addto, совершенно бессмысленно. Любое такое изменение будет отменено командой \begin{document}.

3.7. Перекрёстное цитирование

Мы уже знаем, как изменить способ отображения номера раздела. Интересно было бы также понять, а зачем вообще нумеруют разделы? Совершенно ясно, для чего пронумерованы главы в четырёх томах романа «Война и мир» — они не имеют названий. Впрочем, Жюль Верн обычно находил, что и как назвать, хотя номера тоже писал, вероятно, чтобы не сбиться со счёта. Зачем нумеруют параграфы в учебниках, знает каждый школьник: «А как иначе записать домашнее задание в дневник?» Автор учебника ответит иначе: «Чтобы в параграфе 10 можно было бы сослаться на правило из параграфа 2».

Схема организации печатного документа в LaTeX'е позволяет делать *ссылки* вперед и назад по тексту документа на номера любых пронумерованных объектов. При изменении числа таких объектов ссылки автоматически перенумеруются. Помимо разделов, номера могут иметь рисунки, таблицы, уравнения, записи в списках. LaTeX даёт возможность легко определять новые нумерованные объекты в дополнение к перечисленным. Схема организации автоматических ссылок называется перекрёстным цитированием. Мы поясним её на примере ссылок на номера разделов. Особенности организации ссылок на рисунки и таблицы дополнительно рассматриваются в главе 11, а на цитированную литературу — в главе 13. Ссылки на уравнения (глава 6) и пронумерованные записи в списках (глава 5) организуются совершенно так же, как ссылки на разделы печатного документа.

Чтобы сослаться на какой-либо раздел, необходимо прежде всего пометить его. Метку ставит команда \label. Команда \ref печатает по этой метке номер помеченного раздела, а команда \pageref печатает ссылку на номер страницы, на которую попадает команда \label (помеченный объект может занимать несколько страниц). Итак, достаточно всего трёх команд:

```
\label{key}
\ref{key}
\pageref{key}
```

Здесь *key* — ключ метки, состоящий из любой последовательности букв, цифр и знаков пунктуации[5]. Прописные и строчные буквы различаются. По команде `\label` компилятор запоминает под именем *key* значение счётчика *ctr*, соответствующего *текущему* пронумерованному объекту, а также номер страницы, где расположена команда. Счётчик объявляется текущим (по терминологии LaTeX'а — `ref`-значением) декларацией `\refstepcounter` (раздел 2.9). Она используется командами секционирования и всеми процедурами, которые что-нибудь нумеруют. В процедурах `figure` и `table`, используемых для размещения рисунков и таблиц, текущее `ref`-значение вырабатывает команда `\caption`, которая печатает подписи к рисункам и таблицам. Форма представления ссылки, которую использует команда `\ref`, определяется командой `\the`*ctr*, соответствующей счётчику *ctr*. Номер страницы, печатаемый командой `\pageref`, зависит от определения `\thepage`.

Запоминая метку, компилятор записывает её в файлы с расширением `aux`. Записанная информация из этих файлов считывается командой `\begin{document}` в начале каждой обработки входного файла компилятором. Поэтому она может устареть, если входной файл был изменён. В таком случае компилятор печатает на экране дисплея предупреждение

```
LaTeX Warning: Label(s) may have changed.
Rerun to get cross-references right⁶.
```

Тогда нужно выполнить компиляцию входного файла ещё раз либо проигнорировать предупреждение, если работу предполагается продолжить. Компилятор также печатает предупреждение, если обнаруживает одну и ту же метку в двух или более командах `\label`:

```
Label `...' multiply defined⁷.
```

Область действия декларации `\refstepcounter` подчиняется обычным правилам. Поэтому, чтобы пометить раздел, команда `\label` должна находиться после команды секционирования в любом месте раздела до любой следующей команды секционирования.

В качестве примера покажем, как оформлена ссылка на тот раздел первой главы, где приведён образец входного файла. В первой строке этого раздела во входном файле установлена метка с ключом `c:int/first`:

---

[5] В *key* нельзя использовать русские буквы, поскольку они по сути являются командами. Если всё-таки русские буквы присутствуют в *key*, как случается при обработке старых документов, необходимо загрузить пакет citehack из коллекции T2, причём пакет citehack должен быть загружен *после* пакета babel. В разделе 3.3.7 описана процедура установки нестандартных пакетов, к которым относится пакет `citehack`.

[6] Возможно, изменились метки. Повторите обработку, чтобы получить правильные перекрёстные ссылки.

[7] Метка '`...`' определена повторно.

```
\section{Пример входного файла}\label{c:int/first}
```

Зная ключ метки, легко определить номер раздела и номер страницы, где он начинается:

```
Раздел~\ref{c:int/first} начинается
на странице~\pageref{c:int/first}.
```

Раздел 1.3 начинается на странице 16.

Если меток много, лучше с самого начала выработать правила, которые бы совместили информативность ключей с краткостью. Так, метки разделов мы начинаем с пары символов «c:» (от английского слова chapter — глава), за которой следует аббревиатура, отражающая содержание раздела; метки рисунков — с «f:» (от слова figure), метки таблиц — с «t:» (от слова table).

При интенсивном использовании механизма перекрёстного цитирования большую помощь способен оказать пакет showkeys. Если он загружен, команды \label, \ref, \pageref печатают на полях страницы или между строчками названия ключей, используемых этими командами.

3.7.1. Пакет varioref

Пакет varioref Франка Миттельбаха (Mittelbach, Frank) из коллекции tools расширяет средства перекрёстного цитирования, добавляя к командам \ref и \pageref ряд других, способных печатать более осмысленный текст, нежели просто номера ссылок и страниц, где они расположены:

```
\vref{key}
\vref*{key}
\vpageref[samepage][otherpage]{key}
\vpageref*[samepage][otherpage]{key}
\vrefrange{key1}{key2}
\vpagerefrange{key1}{key2}
\vpagerefrange*{key1}{key2}
```

(varioref)

Действие этих команд зависит от текущего языка, поэтому пакет varioref принимает все опции, перечисленные в разделе 3.6 на с. 76, которые имеются у пакета babel. Мы, естественно, использовали опцию russian. Так как список опций был помечен меткой babeloptions, мы сделали ссылку на него выше следующим образом:

```
...перечисленные в разделе~\vref{babeloptions}, которые имеются...
```

Команда \vref сначала печатает ссылку так же, как \ref{key}, но затем ещё добавляет «на с. 76», «на предыдущей странице» или «на следующей странице», если соответствующая команда \label{key} не находится на текущей странице. В документе, предназначенном для двухсторонней печати (когда на развороте видны сразу две страницы), команда \vref действует ещё более избирательно:

она различает случаи, когда предшествующая или следующая страницы попадают на один разворот со страницей, где печатается ссылка. В этом случае вместо «предыдущей» или «следующей страницы» используется термин «противоположная страница».

Команда `\vpageref` есть вариация `\pageref` со свойствами, подобными команде `\vref`. От последней она отличается, главным образом, тем, что не печатает саму ссылку `\ref`. Если команда `\vpageref{`*key*`}` попадает на ту же страницу, что и соответствующая ей `\label{`*key*`}`, то по случайному выбору она печатает либо «на этой странице», либо «на данной странице».

Наличие необязательных аргументов у `\vpageref` позволяет выполнять ещё более тонкую настройку. Первый необязательный аргумент *samepage* может содержать текст, который будет напечатан, если метка `\label` и ссылка `\vpageref` попадают на одну страницу. Обычно в таком случае известно, находится ли метка до или после ссылки[8]. Тогда текст

```
...см.\ пример \vpageref[выше]{ex:1}...
```

в печатном документе превратится в «см. пример выше», если всё попадет на одну страницу, или в нечто вроде «см. пример на предыдущей странице», если пример и ссылка попадут на разные страницы.

Второй необязательный аргумент у команды `\vpageref` позволяет инвертировать порядок слов. После

```
...см.\ \vpageref[выше пример][пример]{ex:1}, который...
```

в печатном документе появится «см. выше пример», если ссылка на пример и сам пример окажутся на одной странице.

Команда `\vrefrange` предназначена для печати диапазона ссылок. Она имеет два аргумента: key_1 есть метка первой ссылки в диапазоне, а key_2 — метка последней ссылки. Так что если первый и последний рисунки помечены соответственно метками `fig:a` и `fig:c`, то

```
...см.\ рисунки \vrefrange{fig:a}{fig:c}...
```

напечатает «см. рисунки с 3.1 по 3.3 на страницах 23–24» или, если рисунки попадут на соседнюю страницу, «см. рисунки с 3.1 по 3.3 на следующей странице».

Команда `\vpagerefrange` аналогична `\vpageref`, но имеет два аргумента. Если обе ссылки попадают на одну страницу, то `\vpagerefrange` делает то же, что и `\vpageref`. Если ссылки оказываются на разных страницах, то `\vpagerefrange` печатает нечто вроде «на страницах 23-24».

Все перечисленные команды, кроме `\vrefrange`, имеют `*`-версию. Две версии команды различаются способом расстановки пробелов, различие между которыми становится заметным, если команде предшествует не пробел, а другой символ, например скобка:

---

[8] Заметим, что плавающие объекты, которые размещаются в верхней части текущей страницы, могут оказаться расположенными перед тем местом, где они описаны во входном файле, но эта проблему решает пакет flafter (см. раздел 11.1).

`Сравните: \\` `\vref{babeloptions} и \vref*{babeloptions},\\` `(\vref{babeloptions}) и (\vref*{babeloptions}).`	Сравните: 3.6 на с. 76 и3.6 на с. 76, (3.6 на с. 76) и (3.6 на с. 76).

Текст, который печатают команда `\vref` и ей подобные, можно изменить, поскольку он хранится в командах-логосах

`\reftextafter`	`\reftextfaceafter`	
`\reftextbefore`	`\reftextfacebefore`	
`\reftextcurrent`	`\reftextpagerange{`key_1`}{`key_2`}`	(varioref)
`\reftextfaraway{`key`}`	`\reftextlabelrange{`key_1`}{`key_2`}`	

Текст ссылки на предшествующую, но невидимую на развороте страницу печатает команда `\reftextbefore`, а на предшествующую и видимую страницу — команда `\reftextfacebefore` (когда текущая страница имеет нечётный номер, а печатный документ обрабатывается с опцией `twoside`). Аналогичным образом команда `\reftextafter` используется, когда `\label` приходится на следующую невидимую, а `\reftextfaceafter` — на следующую видимую страницу. Команда `\pagevref` использует строку, которая хранится в `\reftextcurrent` и предназначена для замещения номера страницы в том случае, когда метка и ссылка находятся на одной странице. Команда `\reftextfaraway` используется для печати номера страницы ссылки, отличающегося более чем на 1 от номера текущей страницы, а также в том случае, когда страницы нумеруются не арабскими, а, скажем, римскими цифрами (раздел 17.1). Дабы не утомлять Читателя дальнейшими объяснениями, просто приведём пример, который воспроизводит определения всех команд `\reftext`cmd в пакете varioref для русского языка:

```
\renewcommand{\reftextfaceafter}
    {на \reftextvario{противоположной}{следующей} странице}
\renewcommand{\reftextfacebefore}
    {на \reftextvario{противоположной}{предыдущей} странице}
\renewcommand{\reftextafter}{на следующей странице}
\renewcommand{\reftextbefore}
    {на \reftextvario{предшествующей}{предыдущей} странице}
\renewcommand{\reftextcurrent}{на \reftextvario{этой}{данной} странице}
\renewcommand{\reftextfaraway}[1]{на с.~\pageref{#1}}
\renewcommand{reftextpagerange}[2]{на страницах~\pageref{#1}--\pageref{#2}}
\renewcommand{reftextlabelrange}[2]{с~\ref{#1} по~\ref{#2}
```

Здесь использована ещё одна команда, определённая в пакете varioref:

`\reftextvario{`$string_1$`}{`$string_2$`}`	(varioref)

Она вносит разнообразие в текст, который печатают команды `\reftext`cmd, выбирая одну из двух строк `{`$string_1$`}`, `{`$string_2$`}` в зависимости от количества ссылок, уже встретившихся в исходном тексте.

3.7.2. Пакет xr

Пакет xr Дэвида Карлайла (Carlisle, David) из коллекции tools предоставляет возможность делать ссылки на пронумерованные объекты из другого документа. Для этого необходимо загрузить пакет xr и затем (в преамбуле) с помощью декларации

`\externaldocument[prefix]{file}` (xr)

указать имя корневого файла *file* внешнего документа, где содержатся ссылки. Например, если этот файл называется `third.tex`, нужно написать

```
\usepackage{xr}
\externaldocument{third}
```

После этого команды `\ref` и `\pageref` смогут напечатать ссылки на любой объект, помеченный командой `\label` как в текущем документе, так и в `third.tex`. Можно перечислить любое количество внешних документов.

Чтобы исключить ситуацию, когда текущий и внешние документы содержат команды `\label{key}` с одинаковым ключом *key*, можно добавить необязательный аргумент *prefix* к `\externaldocument` и такой же префикс *prefix* к каждой ссылке на внешний документ. Если внести в пример выше следующее изменение

```
\externaldocument[3rd-]{third}
```

то ссылку на раздел в `third.tex`, помеченный при помощи `\label{intr}`, будет печатать команда `\ref{3rd-intr}` (тогда как внутри `third.tex` по-прежнему следует писать `\ref{intr}`).

3.8. Большой документ

На обработку большого документа на маломощном компьютере LaTeX может затратить значительное время. При редактировании одного-двух разделов неразумно перерабатывать снова и снова весь документ, бо́льшая часть которого уже успешно сформатирована. Полезно разделить входной файл на несколько файлов меньшего размера. Вне зависимости от того, сколько используется входных файлов, один из них является *корневым*; это тот файл, чьё имя получает компилятор в качестве входного параметра.

LaTeX предлагает два способа разделения входного файла на части. Начнём с более простого. Команда

`\input{file}`

приводит к тому, что содержимое файла *file* обрабатывается LaTeX'ом точно так же, как если бы оно было переписано в тот файл, который содержит эту команду и помещено точно в то место, где она стоит. Имя файла *file* можно указать вместе с расширением или без него; в последнем случае считывается файл *file*`.tex`. LaTeX

считывает вводимый файл от начала и до конца либо до первой встретившейся команды

```
\endinput
```

Если файл не найден, LaTeX фиксирует ошибку и требует указать имя другого файла. Команда `\input` может находиться в любом месте входного файла. В свою очередь вставляемый файл также может содержать команды `\input`.

Помимо удобства работы с небольшими файлами команда `\input` позволяет легко включать один и тот же текст в несколько документов. Например, корневой файл с текстом нашей книги в момент редактирования данной главы мог бы иметь следующий вид

```
\input{preamble}
\begin{document}
%\input{ch0}   % \chapter{Вместо предисловия}
%\input{ch1}   % \chapter{Пособие для начинающих}
%\input{ch2}   % \chapter{Команды и процедуры}
\input{ch3}   % \chapter{Печатный документ}
  . . .
\end{document}
```

Файл `preamble.tex`, который считывается первым, начинается с `\documentclass` и содержит все необходимые декларации. Его можно использовать в тестовых документах для проверки примеров, которые затем будут включены в книгу. Следующие три команды `\input` «закомментированы», что позволило «отключить» большой фрагмент текста, состоящий из введения и двух первых глав. Оставляя «незакомментированным» какой-либо один файл, можно быстро обработать по частям весь документ.

Однако такой способ имеет недостаток, так как сбивает нумерацию страниц и делает невозможными ссылки на «отключенные» главы. Разумеется, после завершения редактирования всех глав можно убрать знаки комментария `%` и пропустить через компилятор весь текст целиком.

Второй способ дробления входного файла позволяет поддерживать правильную нумерацию страниц и ссылок в ходе всей работы. В этом случае вместо `\input` используется команда

```
\include{file}
```

а в преамбулу необходимо вставить декларацию

```
\includeonly{file-list}
```

где file-list содержит список файлов, перечисленных через запятую. Правило именования файлов теперь иное: расширение имени файла опускается, а считывается файл с расширением `tex`; однако, если он не найден, компилятор не фиксирует ошибку, а печатает предупреждение.

Команда `\include` прежде всего вызывает переход на новую страницу (если текущая страница не пуста), а затем считывает файл *file*. Если файла *file* нет в списке *file-list*, то команда `\include` просто вызывает переход на новую страницу (если текущая страница не пуста).

В отличие от `\input` команда `\include` не должна стоять ранее команды `\begin{document}`, так как вызывает переход на новую страницу, а в преамбуле могут находиться только декларации, которые не производят реальных действий. Чтобы «отключить» какой-нибудь файл, теперь не нужно удалять команду `\include` с его именем; достаточно удалить имя файла из списка *file-list*. Тогда текст файла не будет считан, но вся информация об установленных в нём метках, нумерации страниц, рисунков, таблиц и т. д. будет извлечена из файла *file*`.aux`, поэтому последующая часть входного файла будет обработана так, как если бы *file*`.tex` был действительно считан и обработан. Файл *file*`.aux` обновляется всякий раз, когда файл *file*`.tex` обрабатывается LaTeX'ом, то есть его имя присутствует в списке файлов в `\includeonly`. В следующем разделе рассказано, как вводить список файлов с клавиатуры во время обработки входного файла.

Так как фрагмент текста, считываемый командой `\include`, всегда начинается с новой страницы, эту команду удобно использовать для включения глав, которые, как правило, начинаются с новой страницы. Если обрабатывать файлы по порядку, в котором они перечислены в командах `\include`, то текст всего документа сохранит сквозную нумерацию страниц, разделов, формул и пр. При изменении содержания какой-либо главы следует обработать заново все последующие главы, чтобы восстановить правильную нумерацию перекрёстных ссылок, хотя эту операцию можно отложить до окончания всей работы.

Имеется два ограничения на содержание файла, указанного в аргументе команды `\include`. Во-первых, команды `\include` не могут быть вложены друг в друга, поэтому включаемый файл не может содержать команду `\include`. Во-вторых, внутри включаемого файла нельзя определять новые счётчики при помощи `\newcounter`. Поскольку декларация `\newcounter` имеет глобальную область действия, наиболее подходящим местом для её размещения является преамбула корневого входного файла.

3.9. Условная компиляция

При обработке большого входного файла иногда полезно получать сообщения, чем LaTeX занят в данный момент времени. Для этого имеется команда

⚠ `\typeout{`*msg*`}`

которая выводит сообщение *msg* на экран[9] и в протокол компиляции *name*`.log`. Это может быть простое напоминание о необходимости изменить какой-то фраг-

[9] Для правильного отображения русских букв на экране в момент компиляции компилятор должен быть вызван с ключом `--termainal=oem`, если используется библиотека исполняемых программ MiKTeX.

мент текста, которое легко пропустить, если записать его в виде обычного комментария во входном файле. Например:

```
\typeout{Проверь уравнение!}
```

```
Проверь уравнение!
```

Если в *msg* имеется команда, то она замещается её определением перед тем, как будет высвечена на экране. Чтобы высветить имя такой команды, необходимо защитить её командой `\protect`. И вообще, команды LaTeX'а, стоящие в *msg*, как и команда `\typeout`, помещённая в аргументе другой команды, могут дать неожиданные результаты.

LaTeX, как обычно, заменяет кратные пробелы в *msg* одним и игнорирует пробелы после имени команды. Команда

`\space`

помещённая в *msg*, вставляет один пробел в текст при выводе на экран.

В процессе обработки входного файла можно вводить информацию для компилятора прямо с клавиатуры. По команде

⚠ `\typein[`*cmd*`]{`*msg*`}`

LaTeX высвечивает на экране две строки: в первой строке сообщение *msg*, а во второй `@typein=` (или *cmd*=, если имеется опция `[`*cmd*`]`), после чего ожидает ввода, признаком конца которого является нажатие клавиши ⟨Enter⟩.

Введённый текст подставляется на место команды `\typein`, если она не имеет обязательного аргумента. Например:

```
Мою дочь зовут
\typein{Введи имя}.
```

```
Введи имя

\@typein=
```

Если теперь в ответ на приглашение `\@typein=` ввести `Аня`, то в печатном документе появится фраза «Мою дочь зовут Аня».

При наличии необязательного аргумента *cmd* он должен быть именем команды и начинаться с обратного слеша (см. раздел 2.1); `\typein` определяет (или переопределяет) эту команду в соответствии с вводом с клавиатуры. В комбинации с `\includeonly` команда `\typein` позволяет легко обрабатывать большой печатный документ без изменения корневого файла. Следующий пример показывает, как это сделать:

```
\typein[\file]{Имя файла, шеф!}
\includeonly{\file}
```

```
Имя файла, шеф!

\file=
```

Если теперь в ответ на приглашение `\file=` ввести `ch3`⟨Enter⟩, то выполнится команда `\includeonly{ch3}`.

Аргумент *msg* команд `\typeout` и `\typein` является подвижным, а сами команды хрупкими.

3.10. Оглавление

Работу над печатным документом завершает составление оглавления. Иногда также составляют список рисунков и таблиц. LaTeX полностью автоматизирует эту часть работы. Оглавление, списки рисунков и таблиц печатают соответственно команды

```
\tableofcontents
\listoffigures
\listoftables
```

Они считывают необходимую информацию из файлов, имена которых совпадают с именем входного файла (точнее, корневого входного файла), а расширения определяются из следующей таблицы:

команда:	`\tableofcontents`	`\listoffigures`	`\listoftables`
расширение:	`toc`	`lof`	`lot`

Но как нужная информация попадает в перечисленные файлы? Она записывается туда в момент исполнения команды `\end{document}`, но только в том случае, если исходный текст содержит соответствующую команду `\tableofcontents`, `\listoffigures`, `\listoftables`, а в преамбуле отсутствует декларация

```
\nofiles
```

которая запрещает запись любых служебных файлов. Поскольку содержание оглавления и списков рисунков и таблиц может отставать от текущей версии входного файла, его окончательный вариант необходимо обработать LaTeX'ом дважды, а иногда и трижды, если оглавление и списки помещены в начало печатного документа (например, где-нибудь после заголовка). Однако на последней стадии редактирования в текст вносятся обычно незначительные изменения, не влияющие на содержание оглавления или списка рисунков и таблиц. Всякий раз, когда оглавление, списки рисунков или списки таблиц изменились, LaTeX печатает предупреждение: `Label(s) may have changed.` Тогда нужно обработать входной файл ещё раз либо проигнорировать предупреждение, если работу предполагается продолжить.

В статьях оглавление, списки рисунков и таблиц печатают вслед за основным текстом, а в книгах и отчётах они, как правило, располагаются на отдельных страницах. Соответствующие классы печатных документов учитывают эти традиции.

Информация для оглавления формируется командами секционирования, рассмотренными в разделе 3.5, а информация о рисунках и таблицах — командой `\caption` в процедурах `figure` и `table` (глава 11). Кроме того, такую информацию можно записать с помощью следующих двух команд:

`\addcontentsline{`*ext*`}{`*unit*`}{`*entry*`}` добавляет запись *entry* в файл с указанным расширением *ext*, форматируя её в соответствии со значением аргумента *unit*. Расширение имени файла *ext* может принимать одно из трёх значений: `toc`, `lof`, `lot`. В зависимости от конкретного выбора *ext* второй аргумент *unit* должен иметь следующие значения:

`toc`: `part`, `chapter`, `section` и т. д. (имя команды секционирования без \)

`lof`: `figure`

`lot`: `table`

Текст записи *entry* является подвижным аргументом. Чтобы в оглавление или список ввести запись с номером раздела, рисунка или таблицы, *entry* должен иметь следующую форму:

`\protect\numberline{`*num-unit*`}{`*entry-head*`}`

где *num-unit* — номер, а *entry-head* — текст записи.

`\addtocontents{`*ext*`}{`*text*`}` Добавляет текст (или команды) непосредственно в файл, в который LaTeX записывает оглавление или список рисунков и таблиц. Здесь *ext* по-прежнему означает расширение имени файла, в который следует записать информацию (`toc`, `lof`, `lot`). Второй аргумент *text* является подвижным и должен содержать текст записи.

Команды `\addcontentsline` и `\addtocontents` позволяют редактировать служебные файлы с расширением `toc`, `lof` и `lot`, если необходимо осуществить тонкую настройку, например при ручном регулировании линии переноса на новую страницу части длинной таблицы. Однако лучше такую настройку делать при подготовке окончательной версии печатного документа, используя декларацию `\nofiles` (см. выше), которая предотвращает запись новых версий служебных файлов.

Предисловие к книге часто пишут после того, как завершена бо́льшая часть работы. Иногда «предисловие» называют послесловием и помещают в конце книги. Так или иначе, оформление того и другого может потребовать таких сложных команд, как `\addcontentsline`. Последний пример в данном разделе показывает, как оформлено предисловие к нашей книге.

```
\chapter*{Вместо предисловия}
\addcontentsline{toc}{chapter}{Вместо предисловия}
```

Здесь проблемы создала команда `\chapter*`. Напомним, что `*`-форма команд секционирования не передаёт заголовки разделов в оглавление и колонтитулы.

Поэтому понадобилась команда \addcontentsline, чтобы восполнить этот пробел. Чтобы занести заголовок в колонтитулы нужна ещё команда \markboth; она описана в разделе 17.1.

3.10.1. Параметры настройки

Параметры настройки определяют, разделы какого уровня нужно нумеровать и разделы какого уровня следует записывать в оглавление. Каждый раздел имеет уровень, выражаемый числом. Стандартные классы приписывают секции, т. е. разделу, который начинается с команды \section, уровень 1, подсекции, начинающейся с \subsection,— уровень 2 и т. д. В классе article часть (\part) имеет уровень 0, в классах book и report уровень 0 имеет глава (\chapter), а часть имеет отрицательный уровень −1.

Следующие два счётчика

```
secnumdepth
tocdepth
```

контролируют глубину нумерации разделов. В счётчике secnumdepth хранится уровень младшей секции, заголовок которой имеет номер, а в tocdepth — уровень младшей секции, которая перечисляется в оглавлении. Значение этих счётчиков обычно устанавливают в преамбуле, используя декларацию \setcounter. Например, для этой книги мы выбрали

```
\setcounter{secnumdepth}{2}
\setcounter{tocdepth}{1}
```

В результате нумеруются разделы вплоть до подсекций, но в оглавление попадают только главы и секции.

Другая группа параметров суть команды, хранящие названия оглавления, списка рисунков и таблиц:

```
\contentsname
\listfigurename
\listtablename
```

Рекомендуемый способ переопределения подобных команд в многоязычном документе описан в разделе 3.6.4.

знаки припенания поставлены автором
просьба не трогать
М. Шехтман. Смерть

Глава 4

От буквы до страницы

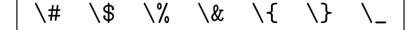
LaTeX рассматривает исходный текст как поток символов, из которых требуется составить строки заданной ширины, а из строк — страницы заданной высоты. На первый взгляд, это совсем несложная задача. Однако LaTeX предъявляет чрезвычайно жёсткие требования к качеству печатного документа. Он считает, что не справился со своей задачей, если какая-нибудь строка в середине абзаца оказалась длиннее соседних на одну десятую пункта, то есть на 0,035 миллиметра. Если Читатель желает узнать об истинных возможностях LaTeX'а, он не должен пропускать эту главу, хотя здесь мы частично повторяем сведения, изложенные во вводной главе.

4.1. Специальные и диакритические знаки

Напомним, что десять символов зарезервированы для служебного пользования:

- `#` обозначает параметр в командах и процедурах,
- `$` обозначает математическую формулу,
- `%` начинает комментарий,
- `&` разделяет колонки в таблицах,
- `{` открывает блок,
- `}` закрывает блок,
- `_` объявляет блок нижним индексом в формуле,
- `\` является признаком команды,
- `^` объявляет блок верхним индексом в формуле,
- `~` запрещает перенос на следующую строку.

Чтобы воспроизвести служебные символы в печатном документе, необходимо использовать специальные команды. Первые семь символов печатаются командами



```
\#  \$  \%  \&  \{  \}  \_
```

которые получаются прибавлением обратного слеша к символу. Обратный слеш \ печатает команда `\textbackslash`, а в математических формулах — команда `\backslash`. Последние два символа принадлежат особому классу знаков, которые называются диакритическими и используются в алфавите некоторых иностранных языков.

Таблица 4.1

Диакритические знаки в текстовом и строковом режимах

ò	`\‘{o}`	õ	`\~{o}`	ǒ	`\v{o}`	o̧	`\c{o}`	ö	`\"{o}`
ó	`\’{o}`	ō	`\={o}`	ő	`\H{o}`	ọ	`\d{o}`	ŏ	`\u{o}`
ô	`\^{o}`	ȯ	`\.{o}`	o̊	`\r{o}`	o̲	`\b{o}`	o͡o	`\t{oo}`
ǫ	`\k{o}`[1)2)]	ô	`\f{o}`[2)]	ȍ	`\C{o}`[2)]	ȍ	`\U{o}`[2)]		

[1)] Имеется в кодировке T1. [2)] Имеется в кодировках T2A, T2B, T2C.

Диакритические знаки и соответствующие им команды приведены в табл. 4.1. Некоторые диакритические знаки имеются только в шрифтах с определённой кодировкой. Это типичная ситуация для команд, которые печатают текстовые символы, поскольку в текстах на разных языках используются разные наборы символов. Подробнее проблемы, связанные с кодировкой, обсуждаются в разделе 16.5. Здесь же достаточно сказать, что по умолчанию используется кодировка `OT1`, а в документах на русском языке (а также украинском и болгарском) — кодировка `T2A`; кодировку `T2A` по умолчанию выбирает пакет babel с опциями `bulgarian`, `russian` или `ukrainian`.

```
Пример фразы на французском языке:
\textbf{‘‘Les machines \‘{a}
\’{e}crire co\^utent cher.’’}
```

Пример фразы на французском языке: **“Les machines à écrire coûtent cher.”**

Фигурные скобки в командах расстановки диакритических знаков являются платой за поддержку стандарта LaTeX’а при обращении к командам TeX’а. Они не обязательны, если предназначены одному символу, а не двум или более. Отметим, что команды `\‘`, `\’`, `\=` из табл. 4.1 в процедуре `tabbing` имеют иной смысл. Там они зарезервированы под команды табуляции, а соответствующие диакритические знаки печатают команды `\a‘`, `\a’`, `\a=` (глава 12).

LaTeX позволяет воспроизводить многие специальные символы иностранных языков. Все семь символов из табл. 4.2 могут использоваться в любом режиме, но команда `\No` доступна только после подключения пакета babel (с опцией `russian`, `bulgarian` или `ukrainian`).

Специальные символы из таблицы 4.3 могут появляться в текстовом и строковом режимах. Если необходимо использовать эти символы в математических формулах, соответствующие команды следует поместить в аргумент команды `\mbox` (раздел 9.1).

```
Сравните: \AA, $\mbox{\AA}$.
```

Сравните: Å, Å.

Таблица 4.2

Команды для специальных символов в любом режиме

†	\dag	§	\S	©	\copyright		
‡	\ddag	¶	\P	£	\pounds	№[1]	\No[2]

[1]Имеется в кодировках TS1, T2A, T2B, T2C. [2]Имеется в пакете babel с опциями bulgarian, russian, ukrainian.

Таблица 4.3

Команды для специальных символов в текстовом режиме

Æ	\AE	æ	\ae	Œ	\OE	œ	\oe
Å	\AA	å	\aa	SS	\SS	ß	\ss
Ł	\L	ł	\l	Ø	\O	ø	\o
Đ	\DH[1]	ð	\dh[1]	Đ	\DJ[1]	đ	\dj[1]
Ŋ	\NG[1]	ŋ	\ng[1]	Þ	\TH[1]	þ	\th[1][2]
¶	\textparagraph			§	\textsection		
†	\textdagger			‡	\textdaggerdbl		
{	\textbraceleft			}	\textbraceright		
<	\textless			>	\textgreater		
␣	\textvisiblespace			•	\textbullet		
∗	\textasteriskcentered			·	\textperiodcentered		
\	\textbackslash			|	\textbar		
№[3][4]	\textnumero[3][4][5]						
‰	\textperthousand[1][3]			‱	\textpertenthousand[1][3]		

[1]Имеется в кодировке T1. [2]Конфликтует с одноимённой командой, определённой пакетом babel с опциями bulgarian, russian, serbian, ukrainian. [3]Имеется в кодировке TS1. [4]Имеется в кодировках T2A, T2B, T2C. [5]Имеется в пакете babel с опциями bulgarian, russian, ukrainian, а также в пакете textcomp.

Таблица 4.4

Команды для специальных символов в текстовом режиме (пакет textcomp)

‚	\textquotestraightbase	„	\textquotestraightdblbase
—	\texttwelveudash	—	\textthreequartersemdash
←	\textleftarrow	→	\textrightarrow
␢	\textblank	$	\textdollar
'	\textquotesingle	∗	\textasteriskcentered
=	\textdblhyphen	/	\textfractionsolidus
0	\textzerooldstyle	1	\textoneoldstyle
2	\texttwooldstyle	3	\textthreeoldstyle
4	\textfouroldstyle	5	\textfiveoldstyle
6	\textsixoldstyle	7	\textsevenoldstyle
8	\texteightoldstyle	9	\textnineoldstyle
⟨	\textlangle	−	\textminus

Продолжение табл. 4.4

〉	\textrangle	℧	\textmho
◯	\textbigcircle	Ω	\textohm
⟦	\textlbrackdbl	⟧	\textrbrackdbl
↑	\textuparrow	↓	\textdownarrow
`	\textasciigrave	⋆	\textborn
⚮	\textdivorced	†	\textdied
🙢	\textleaf	⚭	\textmarried
♪	\textmusicalnote	˷	\texttildelow
⸗	\textdblhyphenchar	˘	\textasciibreve
ˇ	\textasciicaron	˵	\textgravedbl
˝	\textacutedbl	†	\textdagger
‡	\textdaggerdbl	‖	\textbardbl
‰	\textperthousand	•	\textbullet
℃	\textcelsius	$	\textdollaroldstyle
¢	\textcentoldstyle	ƒ	\textflorin
₡	\textcolonmonetary	₩	\textwon
₦	\textnaira	₲	\textguarani
₱	\textpeso	₤	\textlira
℞	\textrecipe	‽	\textinterrobang
⸘	\textinterrobangdown	₫	\textdong
™	\texttrademark	‱	\textpertenthousand
¶	\textpilcrow	฿	\textbaht
№	\textnumero	⁒	\textdiscount
℮	\textestimated	◦	\textopenbullet
℠	\textservicemark	⁅	\textlquill
⁆	\textrquill	¢	\textcent
£	\textsterling	¤	\textcurrency
¥	\textyen	¦	\textbrokenbar
§	\textsection	¨	\textasciidieresis
©	\textcopyright	ª	\textordfeminine
🄯	\textcopyleft	¬	\textlnot
℗	\textcircledP	®	\textregistered
¯	\textasciimacron	°	\textdegree
±	\textpm	²	\texttwosuperior
³	\textthreesuperior	´	\textasciiacute
µ	\textmu	¶	\textparagraph
·	\textperiodcentered	※	\textreferencemark
¹	\textonesuperior	º	\textordmasculine
√	\textsurd	¼	\textonequarter
½	\textonehalf	¾	\textthreequarters
€	\texteuro	×	\texttimes
÷	\textdiv		

Почти все символы из табл. 4.3 доступны во всех шрифтах. Однако при использовании машинописного шрифта Computer Modern возникают проблемы с двумя командами `\l` и `\L`:

```
Сравните: \texttt{\l, \L} и \l, \L.
```
Сравните: ␣l,␣L и ł, Ł.

В этом нет ничего удивительного: обычная пишущая машинка не имеет клавиш ⟨ł⟩ и ⟨Ł⟩.

Некоторые математические символы изредка встречаются в обычном тексте. Такие символы нетрудно вставить в текст в виде формул, но предпочтительнее использовать специальные команды `\text`*cmd*, также собранные в табл. 4.3. Символам •, ·, \, |, { и } из этой таблицы соответствуют команды `\bullet`, `\cdot`, `\backslash`, `\mid`, `\{` и `\}`, применяемые в математических формулах (глава 6). Символ ␣ обозначает пробел и используется в листингах компьютерных программ. Знаки $<$ и $>$ в математических формулах набираются очевидным способом при помощи клавиш, имеющихся на любой клавиатуре, но вне формул символы < и > после компиляции исходного текста в печатный документ превращаются в нечто, зависящее от текущего языка (точнее, от кодировки шрифта). Использование команд `\text`*cmd*, напротив, всегда гарантирует требуемый результат.

Ещё две команды из серии `\text`*cmd*, а именно

```
\textcircled{text}
\textcompwordmark
```

имеют другое назначение. Первая рисует кружок вокруг текста. Например:

```
\textcircled{a}, \textcircled{c}, \copyright.
```
ⓐ, ⓒ, ©.

Другая команда используется для разбиения лигатур. Например, сочетание «fi», набранное при помощи `f\textcompwordmark i`, отличается от «ﬁ», набранного просто как `fi`. Чтобы разбить лигатуру «ﬁ» в слове definition (определение), можно просто написать `def{}inition`, но тогда LaTeX не сможет перенести это слово по слогам, а команда `\textcompwordmark` не запрещает перенос.

Набор текстовых символов значительно расширяется при загрузке пакета textcomp. Соответствующие команды перечислены в таблице 4.4.

Ряд символов можно получить с помощью лигатур. Любая лигатура имеет эквивалентную ей команду. Например, перевёрнутый знак вопроса ¿ можно набрать как лигатуру `?‘`, но только если текущий язык не русский. Команда `\textquestiondown` печатает ¿ вне зависимости от выбора языка. Команды и лигатуры для набора текстовых (т. е. не математических) символов собраны в табл. 4.5, причём лигатуры указаны справа от команды.

4.2. Всё о предложении

Типографские правила путь немного да отличаются в разных странах. Например, в текстах на русском языке принято использовать кавычки в виде «ёлочек»,

Таблица 4.5

Команды для замены лигатур в текстовом режиме

—	`\textemdash`	`---`	–	`\textendash`	`--`
¡	`\textexclamdown`	`!‘`[1)2)]	¿	`\textquestiondown`	`?‘`[1)2)]
‘	`\textquoteleft`	`‘`	’	`\textquoteright`	`’`
“	`\textquotedblleft`	`‘‘`	”	`\textquotedblright`	`’’`
«	`\guillemotleft`[2)3)4)]	`<<`[3)]	»	`\guillemotright`[2)3)4)]	`>>`[3)]

[1)]Имеется в кодировке `OT1`. [2)]Имеется в кодировке `T1`. [3)]Имеется в кодировках `T2A`, `T2B`, `T2C`. [4)]Для других кодировок эмулирована пакетом babel.

тогда как англичане обычно пишут “. . . ”, а немцы „ . . . “. Приглядевшись, можно также заметить, что различается длина тире. «Импортное» тире (a—b) длиннее, чем «отечественное» (а — б), и не должно быть окружено пробелами. Знатоки могут перечислять примеры подобных различий чуть ли не до бесконечности, но мы разберём наиболее известные из них.

4.2.1. Кавычки

В английском языке набор кавычек включает два варианта:

`‘...’`	‘одинарные’
`‘‘...’’`	“двойные”

В русских изданиях кавычки изображаются в виде «ёлочек»:

`<<...>>`	«русские»

Их называют русскими, но с не меньшим основанием своими их также считают французы и немцы. Однако способ набора «ёлочек» с помощью лигатур, т. е. идущих подряд двух символов `<` или `>`, возможен только при использовании русских шрифтов, точнее, кодировок `T2A`, `T2B`, `T2C`. Пакет babel по умолчанию выбирает кодировку `T2A` для русского языка. В иных случаях можно использовать команды из табл. 4.6, где представлены все мыслимые виды кавычек. Пакет babel эмулирует кавычки нужного вида, если их нет в текущей кодировке; это утверждение верно только для команд, печатающих кавычки, но не для лигатур. Для каждого конкретного языка пакет babel может вводить дополнительные команды набора кавычек. Например, для русского языка существует ещё по крайней мере три способа набора «ёлочек»[1]. Однако мы договорились не перегружать Читателя избыточной информацией.

4.2.2. Дефисы и тире

Дефисы и тире набирают при помощи лигатур:

[1] См. `russian.dtx`, а также раздел 3.6.2.

Таблица 4.6

Кавычки

‘	\textquoteleft	`	’	\textquoteright	'
“	\textquotedblleft	``	”	\textquotedblright	''
«	\guillemotleft[1)2)]	<<[2)]	»	\guillemotright[1)2)]	>>[2)]
‹	\guilsinglleft[1)]		›	\guilsinglright[1)]	
„	\quotedblbase[1)2)]	,,[1)2)]	‚	\quotesinglbase[1)]	
"	\textquotedbl[1)2)]				

[1)] Имеется в кодировке T1. [2)] Имеется в кодировках T2A, T2B, T2C.

-	(-)	в составных словах
--	(–)	между числами
---	(—)	между словами

Дефисы и тире встречаются в тексте, но не в математических формулах, где даже длина знака минус отличается от длины дефиса.

```
Сравните: (-), (--) и $(-), (--)$.
```

Сравните: (-), (–) и $(-), (--)$.

Пакет babel с опцией russian вводит дополнительные команды:

\cdash---	(—)	тире в тексте	(babel)
\cdash--~	(—)	тире в составных словах	
\cdash--*	(—)	тире в прямой речи	

Они по-разному расставляют пробелы вокруг тире.

4.2.3. Логосы

Следующие команды предназначены для использования только в текстовом и строковом режимах:

\TeX	печатает аббревиатуру TeX
\LaTeX	печатает аббревиатуру LaTeX
\LaTeXe	печатает аббревиатуру LaTeX 2ε
\today	печатает текущую дату 30 апреля 2004 г.
\ldots	печатает многоточие …

Формат, в котором команда \today печатает текущую дату, определяется классом печатного документа. Пакет babel с опцией russian модифицирует эту команду так, что названия месяцев печатаются по-русски:

```
Эта страница напечатана \today{}
```

Эта страница напечатана 30 апреля 2004 г.

4.3. Горизонтальные пробелы

LaTeX автоматически устанавливает пропорциональное расстояние между словами в строке. При этом не имеет значения, сколько пробелов между словами имеется во входном файле. Случается, впрочем, что LaTeX нуждается в инструкциях о величине пробела между словами. Тогда в распоряжении Читателя имеются следующие команды и декларации:

`\,` вставляет маленький пробел (между одинарными и двойными кавычками);

`\␣` вставляет пробел нормального размера;

`~` вставляет неразрывный пробел нормального размера (например, между словами, которые нельзя отрывать друг от друга);

`\/` вставляет корректирующий пробел при переходе от наклонного или курсивного шрифта (включаемого декларациями `\itshape`, `\slshape`, `\em`) к прямому;

`\@` увеличивает пробел после знака пунктуации в конце предложения; команда ставится перед знаком пунктуации; она необходима, если символ, предшествующий точке, вопросительному или восклицательному знаку, не является строчной буквой или цифрой;

⚠ `\frenchspacing` декларирует подавление дополнительного пробела после знака пунктуации в конце предложения даже в том случае, когда имеется `\@`; включается автоматически при выборе русского языка;

⚠ `\nonfrenchspacing` имеет действие, обратное `\frenchspacing`; действует по умолчанию при выборе большинства иностранных языков.

Перечисленные команды и декларации предназначены для использования в текстовом и строковом режимах, но некоторые из них имеют аналогичное значение и в математическом режиме (глава 6).

Самый маленький пробел `\,` имеет важное применение. Его вставляют в сокращения «т. е.» (`т.\,е.`), «т. п.» (`т.\,п.`), «т. д.» (`т.\,д.`), а также между инициалами и фамилией:

```
Изобретатель трёхфазного тока
М.О.\,Доливо-Добровольский.
```

Изобретатель трёхфазного тока М.О. Доливо-Добровольский.

Одновременно он препятствует переносу на новую строку фамилии, но в следующем разделе мы расскажем, как разрешить переносы в подобных случаях.

Обратим внимание на ещё один важный случай, когда желательно корректировать пробелы между словами. Текст, выделенный курсивом, полезно заканчивать корректирующей командой `\/`:

```
{\itshape Пере}стройка\\
{\itshape Пере\/}стройка
```

*Пере*стройка
*Пере*стройка

Чтобы не заботиться о коррекции, следует использовать команды, а не декларации переключения формы шрифта. Они делают коррекцию автоматически:

`\textit{Пере}стройка\\`	*Пере*стройка
`\textit{Пере\/}стройка`	*Пере*стройка

Как заправский наборщик в типографии, LaTeX может автоматически увеличивать пробел после точки, заканчивающей предложение. Чтобы продемонстрировать это, мы включили режим `\nonfrenchspacing` для данного параграфа:

```
\begin{nonfrenchspacing}
Как заправский наборщик в типографии, \LaTeX\ может автоматически
. . .
сколь аккуратен может быть \LaTeX{}, если его хорошо попросить!
\end{nonfrenchspacing}
```

Человеку нетрудно найти, какая точка означает конец предложения, а какая используется для иных целей. Не мудрствуя лукаво, LaTeX считает, что точка фиксирует конец предложения, если она не следует за прописной (заглавной) буквой. Однако точка может стоять и в середине предложения, как в сокращении «см. рис. 1», и после заглавной буквы в конце предложения. В первом случае правильное расстояние между словами устанавливается, если *после* точки поставить неразрывный пробел `~` или `\␣`, где ␣ обозначает пробел в исходном тексте:

`см. рис. 1\\`	см. рис. 1
`см.\ рис.~1`	см. рис. 1

Что касается точки, восклицательного или вопросительного знака после заглавной буквы, LaTeX будет считать, что они заканчивают предложение, если непосредственно *перед* ними стоит команда `\@`:

`I + II = III\@. Верно!`	I + II = III. Верно!
`I + II = III.  Неверно!\\`	I + II = III. Неверно!

Различие, как видим, мизерное, но этот пример показывает, сколь аккуратен может быть LaTeX, если его хорошо попросить!

Другая причина для перехода на «ручное» управление величиной пробелов — желание достичь визуальных эффектов. Команда `\␣` вставляет один дополнительный пробел (если один пробел перед ней уже есть!):

`А и Б сидели на трубе\\`	А и Б сидели на трубе
`А \ и \ Б \ сидели на ...`	А и Б сидели на трубе

Команда `\\` здесь вызывает принудительный переход на новую строку, а не вставку двух дополнительных пробелов. Чтобы вставить два пробела, нужно использовать две команды `\␣`:

`А \ \ и \ \ Б \ \ сидели`	А и Б сидели

Команды

```
\hspace{len}
\hspace*{len}
```

могут вставлять пробел произвольной длины *len*. В следующем примере `\hspace` добавляет пробел длиной 7 миллиметров к имеющимся до и после неё:

```
A\hspace{7mm}и Б\\
A\hspace{7mm} и Б\\
A \hspace{7mm} и Б
```

А и Б
А и Б
А и Б

Длина может быть явной или командной, положительной или отрицательной (раздел 2.10):

```
пример наложения \hspace{-7mm}
///// текста.
```

пример наложения///// текста.

Если `\hspace` попадает на край строки, то пробел удаляется, однако *-форма команды создаёт неудаляемый пробел:

```
Сравните:\\
\hspace{7mm} А и Б сидели\ldots\\
\hspace*{7mm} А и Б сидели\ldots
```

Сравните:
А и Б сидели...
А и Б сидели...

Для команды `\hspace{\fill}`, вставляющей пробел бесконечно растяжимой длины `\fill`, имеется аббревиатура

```
\hfill
```

Две такие команды в одной строке имеют равную длину:

```
A\hfill и\hfill б
```

А и Б

Команды

```
\hrulefill
\dotfill
```

заполняют свободный промежуток в строке горизонтальной линией или точками:

```
пробел из \hrulefill\ линий,\\
пробел из \dotfill\ точек.
```

пробел из ________________ линий,
пробел из .... точек.

4.4. Как LaTeX делает строки

LaTeX имеет простой, но эффективный алгоритм разбиения слов на слоги. Обычно он находит 80–90% всех правильных переносов (см. [2]), что вполне достаточно для практических целей. Осуществляя автоматическое разбиение текста на строки, LaTeX хорошо справляется со своей работой. Однако иногда ему всё-таки требуется помощь. Например, он не способен определить логически связанные сочетания слов, которые желательно располагать на одной строке. Скажем, обращение «гр. Шариков» выглядит странно, если «гр.» заканчивает одну строку, а «Шариков» начинает следующую. Символ ~ (тильда), вставленный между соседними словами, создаёт пробел нормального размера, на котором LaTeX никогда не разорвёт строку:

```
\LaTeX~никогда, никогда
не~~ра~зор~вёт~~стро~ку
```

LaTeX никогда, никогда не ра зор вёт стро ку

Подобным свойством, как мы знаем, обладает и самый маленький пробел `\,`.

Слово нежелательно переносить по слогам, если оно по сути является символом, как идентификатор в компьютерной программе. Команда `\mbox` (раздел 9.1) запрещает LaTeX'у переносить её аргумент по частям со строки на строку. В следующем примере LaTeX вынужден сделать большие пробелы между словами, так как он обязан разместить слово «идентификатором» целиком в одной строке:

```
Слово \textsf{FORTRAN} является
\mbox{идентификатором}.
```

Слово FORTRAN является идентификатором.

Использование большого количества связующих пробелов `~` или команд `\mbox` может затруднить поиск подходящего места для переноса. Так, в предыдущем примере LaTeX сообщает о своих затруднениях, печатая на экране дисплея предупреждение о незаполненном горизонтальном боксе с указанием, что источник проблем находится в строках 707–709 входного файла:

```
Underfull \hbox (badness 10000) in paragraph at lines 707--709
Слово \T2A/cmss/m/n/10 FORTRAN \T2A/cmr/m/n/10 яв-ля-ет-ся []
```
[2]

Из этого сообщения следует, что LaTeX не разбивал слово FORTRAN на слоги (так как это иностранное слово), а в слове «является» он нашёл три переноса из трёх: `яв-ля-ет-ся`. Немного изменив пример, мы обнаружим, что слово «идентификатором» вылезло за правую границу страницы:

```
Слово \textsf{WHILE} является
\mbox{идентификатором}.
```

Слово WHILE является идентификатором.

В этом случае LaTeX печатает предупреждение о переполненном горизонтальном боксе:

```
Overfull \hbox (12.18564pt too wide) in paragraph at lines 735--737
Слово \T2A/cmss/m/n/10 WHILE \T2A/cmr/m/n/10 яв-ля-ет-ся []
```
[3]

Здесь `[]` символизируют команду `\mbox` вместе с её аргументом. Если в декларации `\documentclass` была использована опция `draft`, то строка, вызвавшая затруднения, в печатном документе будет помечена справа закрашенным прямоугольником. По умолчанию используется опция `final`, и строка не метится.

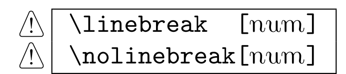
Получив предупреждение о незаполненном или переполненном горизонтальном боксе, полезно изменить текст так, чтобы его фрагмент, вызвавший затруднение, передвинулся внутрь строки. Однако существуют и другие способы разрешения этой проблемы: можно регулировать разбиение строк, переносы и режим форматирования абзацев.

[2] Незаполненный горизонтальный бокс (разрежение 10 000) в абзаце в строках 707–709. Слово (переключение шрифта) FORTRAN (переключение шрифта) яв-ля-ет-ся.

[3] Переполненный горизонтальный бокс (шире на 12.18564pt) в абзаце в строках 735–737. Слово (переключение шрифта) WHILE (переключение шрифта) яв-ля-ет-ся.

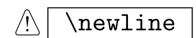
4.4.1. Разрыв строк

LaTeX позволяет управлять разбиением строки между словами.

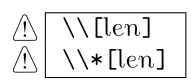


Команда \linebreak поощряет, а команда \nolinebreak препятствует прерыванию строки между словами. Опция *num* должна быть целым числом от 0 до 4. Чем больше значение *num*, тем сильнее поощрение или запрещение. Значение по умолчанию равно 4 и действует как безусловный приказ. Значение 0 равносильно отсутствию команды и не влияет на прерывание строки. Если использование команд \linebreak и \nolinebreak приводит к чрезмерно большим пробелам между словами, LaTeX генерирует предупреждение Underfull \hbox. Команды \linebreak и \nolinebreak хрупкие, они могут использоваться только в текстовом режиме.

Команда



начинает новую строку без выравнивания правого края текущей строки. Она может использоваться только в текстовом режиме и только внутри абзаца, так как вызывает предупреждение Underfull \hbox и дополнительный вертикальный пробел, если стоит в конце абзаца, или сообщение об ошибке, если стоит между абзацами. Команды



ведут себя похожим образом, когда используются в текстовом режиме, причём *-форма команды \\ препятствует переносу новой строки на следующую страницу. Обе команды открывают новую строку и вставляют дополнительный вертикальный пробел длины *len* перед ней. По умолчанию дополнительный пробел не вставляется. Не следует использовать подряд несколько команд \\. Вместо этого в необязательном аргументе нужно указать дополнительное расстояние между строками. Например, \\[7mm].

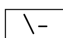
Возможно, Читатель удивится, зачем нужна команда \newline, если есть \\. Отвечаем: она не нужна и поддерживается только по аналогии с командой \newpage, которая начинает новую страницу. В отличие от \newline команды \\ и * могут использоваться не только в текстовом режиме, но и в аргументах некоторых команд, где уместно делать несколько строк (например, в \author, \title, \date, \chapter, \section и т. д.), а также в теле некоторых процедур (мы ещё не рассказывали о таких процедурах, как array, eqnarray, tabular, tabbing, quote, itemize и др.).

Команды \\ и *, как и \newline, хрупкие. Поэтому их нельзя использовать в подвижном аргументе, не защитив при помощи \protect. Это обстоятельство

раньше сильно досаждало любителям длинных заголовков, так как один из аргументов команд секционирования является подвижным (раздел 3.5). Однако теперь команды `\\` и `\\*` в аргументах команд секционирования ведут себя как устойчивые, и там их можно не защищать.

4.4.2. Перенос по слогам

LaTeX не считает возможным переносить слово по слогам, если перед ним стоит маленький пробел `\,`. Переносы также не применяются к сложным словам, написанным через дефис. Однако в подобных случаях можно явно указать место переноса командой



`\-`

Команда `\-` одновременно запрещает все иные варианты переноса в слове, куда она вставлена. Поэтому в следующем примере мы указали все варианты переноса:

```
Изобретатель
М.О.\,До\-ливо-Доб\-ро\-воль\-ский.
```

Изобретатель М.О. Доливо-Доброволь-ский.

Наряду с `\-` есть более гибкая команда

`\discretionary{`$\mathfrak{text}_1$`}{`$\mathfrak{text}_2$`}{`$\mathfrak{text}_3$`}`

Она указывает компилятору, как переносить строку текста в данном месте. Аргумент $\mathfrak{text}_1$ есть текст, который останется на текущей строке; $\mathfrak{text}_2$ — текст, с которого начнётся новая строка; $\mathfrak{text}_3$ — текст, который будет напечатан, если слово целиком уместится в одной строке. Команду `\discretionary` обычно используют, чтобы удалить знак переноса:

```
\discretionary{Река}{Москва}{Москва-река},
\discretionary{река}{Москва}{Москва-река}.
```

Москва-река, река Москва.

Если какое-то слово встречается очень часто, можно вставить в преамбулу декларацию

`\hyphenation{`$\mathfrak{words}$`}`

Она информирует LaTeX, как переносить слова, записанные в $\mathfrak{words}$ (через пробелы), во всех случаях, когда явно не использована команда `\-`. Допустимые точки переноса указываются символом `-`. Например, если кто-то считает неправильным перенос после первого слога в слове «является», он может запретить его во всём документе:

```
\hyphenation{явля-ет-ся явля-ют-ся поч-тальон}
```

LaTeX не переносит слова, набранные машинописным шрифтом или несколькими шрифтами, а также слова, содержащие цифры, скобки или буквы из другого алфавита. Команда `\-` помогает и в этом случае.

4.4.3. Режим форматирования абзацев

LaTeX позволяет повлиять на режим форматирования целых абзацев. Декларации

```
\sloppy
\fussy
```

изменяют способ обрезания строки. Декларация `\fussy`, действующая по умолчанию, препятствует образованию слишком больших пробелов между словами, но разрешает словам вылезать за правую границу страницы, если не найдена подходящая точка для переноса. Декларация `\sloppy` почти всегда обрывает строку в точности на правом краю, но при этом могут образоваться очень длинные промежутки между словами. В этом случае LaTeX выдаст предупреждение `Underfull \hbox` о незаполненном горизонтальном боксе. Область действия декларации `\sloppy` определяется следующим образом. LaTeX выполняет разбиение на строки, когда доходит до конца абзаца (то есть до пустой строки) или до процедуры, начинающей печать с новой строки (примером служат процедуры, описанные в главе 5). Следовательно, форматирование абзаца осуществляется в соответствии с декларацией, действующей на этот момент. В остальном область действия декларации `\sloppy` определяется как обычно. Она простирается до декларации `\fussy`, возвращающей LaTeX в обычный режим, или же может быть ограничена фигурными скобками.

Декларации `\sloppy` соответствует процедура `sloppypar`:

```
\begin{sloppypar}   pars   \end{sloppypar}
```

Аргумент *pars* состоит из одного или нескольких полных абзацев, где будет действовать декларация `\sloppy`.

4.5. Всё об абзаце

Абзац заканчивается одной или несколькими пустыми строками, в которых не должно стоять ничего, даже символ `%`. Пустая строка не должна появляться там, где переход к новому абзацу по сути дела невозможен, в частности в математических формулах (глава 6) и в аргументах команд секционирования (раздел 3.5). LaTeX имеет три команды, регулирующие форматирование абзацев:

`\noindent` подавляет отступ в начале абзаца, перед которым она стоит;

`\indent` производит стандартный отступ в начале абзаца; используется для добавления отступа в тех случаях, когда он почему-либо отсутствует;

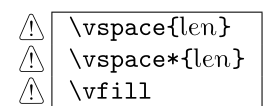
`\par` эквивалентна пустой строке; часто используется в определениях новых команд и процедур.

Количество пустых строк или команд `\par` не имеет значения. Также не имеет значения, с какой позиции начинается первая строка абзаца; LaTeX вставит в неё отступ фиксированной длины. Длина отступа задана командой `\parindent` (раздел 17.2) и зависит от класса печатного документа. Команда `\indent` безвредна, если стоит в начале абзаца, который и без неё начинается с отступа. Отступ имеет каждый абзац, за исключением самого первого после названия главы, раздела, подраздела и т. д., то есть после команды секционирования (раздел 3.5). Так определено стандартными классами LaTeX'а. Команда `\noindent` отменяет отступ, а `\indent`, наоборот, вставляет, но даже она бессильна перед магией первого абзаца. Чтобы LaTeX сделал стандартный отступ в первом абзаце этого раздела, пришлось вставить команду `\hspace*{\parindent}`.

Более кардинальное решение предлагает пакет indentfirst из коллекции tools. Он устанавливает, что первые абзацы (как и все другие) в любом разделе начинаются с отступа.

4.6. Вертикальные пробелы

Вертикальные пробелы формируются почти как горизонтальные. Команды



```
\vspace{len}
\vspace*{len}
\vfill
```

действуют по аналогии со своими «собратьями на букву h» из раздела 4.3.

Как Читатель уже догадался, команда `\vspace` вставляет вертикальный пробел заданной длины. Обычно она используется для вставки дополнительного пробела между абзацами. Если команда появляется внутри абзаца, то вертикальный пробел вставляется после заполнения текущей строки. В данном абзаце сразу

после точки, заканчивающей предыдущее предложение, стоит команда `\vspace {12pt}`, которая вставила вертикальный пробел величиной 12 pt. Если пробел приходится на начало или конец страницы, то он пропадает. Однако команда `\vspace*` вставляет пробел даже в этом случае.

Команда `\vfill` есть аббревиатура `\vspace{\fill}`. Она сдвигает следующий за ней текст в нижнюю часть страницы, вставляя вертикальный пробел бесконечно растяжимой длины `\fill`. Этот пробел исчезает, если попадает на край страницы. Если он всё-таки нужен, следует использовать команду `\vspace*{\fill}`, которая вставляет неудаляемый пробел.

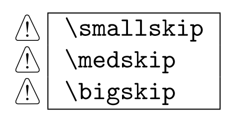
Команда

```
\addvspace{len}
```

не имеет аналогов в разделе 4.3. Она добавляет вертикальный пробел высотой *len*, но если пробел уже был поставлен в этом месте (например, предыдущей командой `\addvspace`), то он не увеличивается более, чем необходимо для создания

вертикального пробела высоты *len*. Обычно `\addvspace` используется для вставки дополнительного пробела над или под процедурами LaTeX'а, образующими новый абзац. Команда может быть использована только в текстовом режиме между абзацами, т. е. после пустой строки или после команды `\par`.

Следующие три команды



```
\smallskip
\medskip
\bigskip
```

вставляют вертикальные пробелы с высотой, предопределённой выбранным классом печатного документа. Высота пробелов задана растяжимыми длинами

```
\smallskipamount    \medskipamount    \bigskipamount
```

соответственно, причём `\big`*cmd* в 2 раза больше, чем `\med`*cmd*, и в 4 раза больше, чем `\small`*cmd*. Заметим, что вставка предопределённых вертикальных пробелов `\bigskip`, `\medskip` и `\smallskip` предпочтительнее явного задания величины пробела, так как позволяет выдержать однородность форматирования текста на всём протяжении печатного документа.

Формат печатной страницы определяется несколькими десятками горизонтальных и вертикальных размеров. Изменение формата страницы достигается переопределением ряда параметров, описанных в разделе 17.2. Например, *интерлиньяж*, т. е. интервал между строками, определяется величиной параметра `\baselinestretch`. Данный абзац напечатан примерно через 0,5 интервала. Перед ним стоит «хитрая» комбинация команд

```
\renewcommand\baselinestretch{0.75}\normalsize
```

а после него —

```
\renewcommand\baselinestretch{1}\normalsize
```

так как реальное изменение интерлиньяжа происходит только при переключении размера шрифта. Чтобы получить полуторный интервал, нужно установить значение `\baselinestretch` на уровне 1.25, а удвоение интерлиньяжа происходит при значении 1.66. Есть также более простой способ изменения интелиньяжа при помощи команды `\linespread`, описанной в разделе 16.3.

Следующий раздел рассказывает, как LaTeX делит текст на страницы.

4.7. Как LaTeX делает страницы

LaTeX так же щепетилен в деле разбиения текста на страницы, как и на строки. Он старательно избегает ситуаций, когда заголовок раздела печатается на одной странице, а текст начинается на другой. Однако иногда LaTeX всё-таки не находит подходящего места для начала новой страницы. В таких случаях он обычно создаёт незаполненную страницу, помещая на неё меньше текста. В зависимости от того, какая из деклараций

```
\flushbottom
\raggedbottom
```

действует на момент завершения форматирования страницы, LaTeX либо увеличивает вертикальные промежутки между абзацами (`\flushdbottom`), либо формирует укороченную страницу (`\raggedbottom`). В первом случае LaTeX предупреждает о своих затруднениях сообщением

`Underfull \vbox ...`[4]

Во втором случае печатный документ надо проверить на наличие укороченных страниц. При двусторонней печати, когда действует опция `twoside` (в классе `book` она используется по умолчанию, см. раздел 3.2) стандартные классы включают режим `\flushbottom`, а при односторонней печати — режим `\raggedbottom`.

Команды

⚠ `\pagebreak[`*num*`]`
⚠ `\nopagebreak[`*num*`]`

помогают при необходимости регулировать разбиение текста на страницы. Они аналогичны командам разбиения на строки, описанным в разделе 4.4. Команда `\pagebreak` поощряет, а `\nopagebreak` препятствует переходу на следующую страницу в зависимости от величины параметра *num*, который может изменяться от 0 до 4. Чем больше его значение, тем сильнее поощрение или запрещение. Значение по умолчанию равно 4 и действует как безусловный приказ. Значение 0 равносильно отсутствию команды. Если эти команды стоят между абзацами, то исполняются немедленно, а если внутри абзаца, то после заполнения текущей строки.

Команды `\pagebreak` и `\nopagebreak` предназначены для применения в текстовом режиме. LaTeX игнорирует их, когда они используются в строковом режиме. Команда `\nopagebreak` не действует, если какая-нибудь другая команда явно разрешает разорвать страницу в данном месте.

Если `\pagebreak` находится в области действия декларации `\flushbottom` и создаёт слишком большие вертикальные пробелы, LaTeX предупреждает об этом сообщением `Underfull \vbox`. То же самое может случиться при попытке запретить переход на новую страницу посредством `\nopagebreak`. Гораздо реже встречается предупреждение

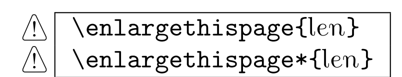
`Overfull \vbox`[5]

LaTeX считает, что лучше оставить много пустого пространства на странице, чем позволить тексту выйти за нижнюю границу, определённую классом печатного документа.

Иногда LaTeX так настойчиво обрезает страницы в определённом месте, что даже команда `\nopagebreak` бессильна остановить его. В этом случае полезно

[4] Незаполненный вертикальный бокс. . .
[5] Переполненный вертикальный бокс.

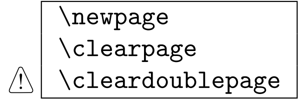
удалить или добавить вертикальные пробелы на «плохой» странице при помощи команд из предыдущего раздела. Если и это не помогает, тогда следует увеличить (или уменьшить) высоту страницы при помощи одной из команд



```
\enlargethispage{len}
\enlargethispage*{len}
```

Они указывают, на какую величину len LaTeX может увеличить высоту текущей страницы. Параметр len должен быть нерастяжимой длиной и может быть отрицательным. Из двух команд вторая (со звездочкой) сильнее первой, так как она пытается ещё и максимально сжать все вертикальные пробелы, имеющиеся на странице. При двусторонней печати полезно обе страницы разворота увеличивать одинаково, чтобы их разная высота не бросалась в глаза. Однако в любом случае подгонку высоты страниц следует отложить до полного окончания работы над текстом печатного документа.

Команды



```
\newpage
\clearpage
\cleardoublepage
```

обрезают текущую страницу и начинают новую.

По аналогии с `\newline`, начинающей новую строку, команда `\newpage` формирует укороченную страницу вне зависимости от декларации `\flushbottom`.

Две другие команды действуют аналогично `\newpage`, но, в отличие от неё, предварительно печатают на отдельных страницах плавающие объекты (глава 11), то есть таблицы и рисунки, ждущие своей очереди на размещение. При односторонней печати команды `\clearpage` и `\cleardoublepage` эквивалентны друг другу. При двусторонней печати команда `\cleardoublepage` в случае необходимости добавляет одну пустую страницу так, чтобы новая страница имела нечётный номер.

При печати в две колонки (когда действует опция `twocolumn`) `\newpage` завершает текущую колонку, а не страницу, тогда как `\clearpage` и `\cleardoublepage` обрезают страницу, производя при необходимости пустую правую колонку.

Все три команды сами по себе не вставляют пустую страницу, поэтому несколько таких команд подряд эквивалентны одной команде. Чтобы образовать чистую страницу, надо на ней поместить невидимый текст, хотя бы с помощью `{ }`. Команды `\newpage`, `\clearpage`, `\cleardoublepage` могут применяться только в текстовом режиме, но не внутри парбоксов (раздел 9.2).

Если Читатель ещё не догадался, как поместить абзац посередине отдельной страницы, советуем перечитать предыдущий раздел. Ответ должен быть примерно таким:

```
\newpage\vspace*{\fill} ... \vspace*{\fill}\newpage
```

Номер текущей страницы хранится в счётчике page, а печатает номер этой страницы команда \thepage. Однако попытка напечатать номер страницы в тексте при помощи \thepage может дать результат, несколько отличающийся от истинного номера страницы, печатаемого в колонтитулах. Если следующий пример попадёт на начало страницы, то почти наверняка \thepage напечатает номер на 1 меньше истинного.

```
Это пример на странице \thepage.
```

Это пример на странице 111.

Причина аномального поведения счётчика page заключается в том, что при выборе места для разбиения текста на страницы LaTeX держит в своей памяти некоторый «излишек» текста. Истинный номер страницы определяется в момент, когда она записывается в файл печатного документа (dvi-файл), а команда \thepage исполняется несколько раньше, в момент форматирования текста.

Существует несколько способов гарантированного получения правильного номера страницы. Лучше всего использовать механизм перекрёстного цитирования (раздел 3.7). Перекрёстное цитирование всегда даёт правильную информацию о ссылке, так как номер нужной страницы определяется одновременно с окончанием форматирования. Поэтому предыдущий пример лучше переделать следующим образом:

```
Это пример на странице
\label{ThisPage}\pageref{ThisPage}.
```

Это пример на странице 111.

4.8. Всё о подстрочном примечании[6]

Подстрочное примечание (сноску) печатает команда

⚠ \footnote[*num*]{*text*}

где *text* — текст сноски, а *num* — её номер, который должен быть положительным числом даже в том случае, когда сноска маркируется буквами или подстрочными символами. Если необязательный аргумент *num* опущен, то сноске присваивается очередной порядковый номер, который хранится в счётчике footnote (для министраниц — mpfootnote). Все стандартные классы, кроме book и report, определяют, что во всём документе применяется единая нумерация сносок. В классах book и report сноски нумеруются отдельно в пределах каждой главы. При удалении или добавлении сноски все оставшиеся перенумеровываются автоматически. Способ маркировки определяется классом печатного документа; по умолчанию сноски нумеруются арабскими цифрами. О том, как изменить маркировку, рассказывает раздел 2.9, где обсуждаются общие свойства счётчиков.

Стандартные классы устанавливают, что сноски печатаются в нижней части страницы, а не в конце печатного документа. При печати в две колонки (при выборе опции twocolumn) подстрочные примечания размещаются под соответству-

[6] Это пример подстрочного примечания к заголовку раздела.

ющей колонкой. Пакет ftnright из коллекции tools размещает все подстрочные примечания под правой колонкой.

Возможности команды `\footnote` обеспечивают большинство потребностей. Напечатать подстрочное примечание можно так:

```
\ldots можно так:
<<Бди!!>>\footnote{\emph{Козьма
Прутков}. Собр. соч. СПб., 1848.}
```

…можно так: «Бди!!»[1]

[1] *Козьма Прутков.* Собр. соч. СПб., 1848.

Маркёр подстрочного примечания печатается в точности там, где стоит обратный слеш `\`, начинающий команду `\footnote`. Поэтому между «Бди!!» и командой `\footnote` в данном примере нет пробела — иначе пробел появился бы и перед маркёром.

Команда `\footnote` может использоваться в текстовом режиме, но не внутри бокса, за исключением процедуры `minipage` (раздел 9.2). В `minipage` возможности команды `\footnote` даже расширяются: там она может использоваться во всех режимах, как показывает следующий пример:

```
\begin{minipage}{...}
\[ \fbox{А\footnote{Аня.}
 + Б\footnote{Боря.}
 = любовь} \]
\end{minipage}
```

$$\boxed{\text{А}^a + \text{Б}^b = \text{любовь}}$$

[a] Аня.
[b] Боря.

Забегая немного вперёд, поясним, что команды `\[` и `\]` метят начало и конец математической формулы, которую LaTeX должен напечатать в центре отдельной строки (глава 6), а команда `\fbox` рисует рамку (раздел 9.2).

Указанное ограничение на использование `\footnote` только в текстовом режиме можно обойти при помощи пары команд

```
\footnotemark[num]
\footnotetext[num]{text}
```

где *num* и *text* имеют прежний смысл.

Первая из них печатает маркёр сноски, но не создаёт саму сноску; она может использоваться в любом режиме. Если в `\footnotemark` опция *num* опущена, то используется счётчик `footnote`[7]: сначала его значение увеличивается на 1, а затем печатается в качестве маркёра подстрочного примечания.

Вторая команда печатает текст подстрочного примечания подобно команде `\footnote`, но не генерирует маркёр и не наращивает значение счётчика `footnote`. Следующий пример показывает, как следует вставлять подстрочное примечание в бокс:

```
\[\fbox{А\footnotemark + Б\footnotemark = любовь}\]
\addtocounter{footnote}{-1}\footnotetext{Аня.}
\addtocounter{footnote}{+1}\footnotetext{Боря.}
```

[7] Даже в `minipage`!

«Результат превосходит ожидание»[8]:

А[9]+ Б[10]= любовь

Все три команды \footnote*cmd* хрупкие. Чтобы сделать подстрочное примечание к заголовку данного раздела, была использована команда \section с необязательным аргументом:

```
\section[Всё о...]{Всё о...\footnote{Это пример...}}
```

При его отсутствии подвижным стал бы обязательный аргумент. Тогда команду \footnote в обязательном аргументе команды \section следовало бы защитить командой \protect[11].

4.8.1. Параметры настройки

\footnotesep — высота страты, т. е. невидимой вертикальной линии (раздел 9.3), которую LaTeX вставляет в начало каждого подстрочного примечания, чтобы создать вертикальный пробел между подстрочными примечаниями. Она может быть изменена в любом месте при помощи \setlength (раздел 2.10); используется то значение, которое было установлено на момент исполнения команды \footnote или \footnotetext.

\footnoterule — команда, которая рисует линию, отделяющую подстрочные примечания от основного текста. Команда используется в текстовом режиме между абзацами (внутренняя вертикальная мода TeX'а). Печатный вывод, производимый этой командой, должен эмулировать объект нулевой высоты, поэтому команда должна вставлять отрицательный вертикальный пробел, чтобы компенсировать место, занятое под линию. Команда может быть переопределена в любом месте посредством \renewcommand (раздел 7.1); используется то определение, которое действует на момент завершения форматирования страницы. См. также раздел 17.2.

[8] Из рекламы одной фирмы.
[9] Аня.
[10] Боря.
[11] Результатом стало бы появление маркёра подстрочного примечания в оглавлении и колонтитулах, что не входило в наши намерения.

Вижу я, что небо небогато,
Но про землю стоит говорить.
Н. Тихонов. Орда

Глава 5

Форматирование абзацев

Общее свойство всех процедур форматирования абзацев *env* состоит в том, что `\begin{`*env*`}` начинает печатать текст с новой строки. Текст, следующий за процедурой, то есть за командой `\end{`*env*`}`, также начинается с новой строки. Однако абзацный отступ вставляется только в том случае, если имеется пустая строка вслед за `\end{`*env*`}`. Правда, после закрытия подряд нескольких вложенных процедур, а также при наличии закрывающей фигурной скобки после `\end{`*env*`}` отступ в начале абзаца всё-таки может появиться даже при отсутствии пустой строки. Такой аномальный отступ может быть устранён при помощи команды `\noindent` (раздел 4.5).

5.1. Позиционирование текста

`\begin{center}`	...	`\end{center}`
`\begin{flushleft}`	...	`\end{flushleft}`
`\begin{flushright}`	...	`\end{flushright}`

Процедура `center` используется для центрирования строк на странице. Она полезна для создания заголовков:

```
Это предшествующий текст.
\begin{center}
{\large \em А. Милн}\\[4pt]
Винни Пух\\ и все остальные
\end{center}
Этот текст следует сразу за
процедурой \texttt{center}.
```

Это предшествующий текст.

А. Милн
Винни Пух
и все остальные

Этот текст следует сразу за процедурой `center`.

Каждая новая строка здесь начинается с команды \\.

По умолчанию LaTeX выравнивает текст по формату (т. е. по правой и левой границам одновременно), варьируя пробелы между словами. Напротив, процедуры `center`, `flushleft` и `flushright` устанавливают равные промежутки между словами, выравнивая строки, соответственно, по центру, левому или правому краю страницы:

```
\begin{flushright}
здесь слова\\ смещены\\ вправо
\end{flushright}
```

здесь слова
смещены
вправо

В процедурах `center`, `flushright` и `flushleft` часто используют команду перехода на новую строку `\\`. Если позволить LaTeX'у самому разбивать текст на строки, то, например, процедура `flushleft` напечатает текст с «рваной» правой границей:

```
\begin{flushleft}
Форматирование текста с выравниванием
по левой границе часто используют для
печати в узкой колонке.
\end{flushleft}
```

Форматирование текста с
выравниванием по левой границе
часто используют для печати в узкой
колонке.

Процедуры позиционирования работают, используя определённые декларации, которые изменяют способ форматирования абзацев, причём имена соответствующих деклараций и процедур совпадают. Например, процедуре `center` соответствует декларация `\center`. Она, как и одноимённая процедура, вставляет вертикальный пробел, как перед началом абзаца. Однако иногда такой пробел не желателен. На этот случай припасены декларации `\centering`, `\raggedright` и `\raggedleft`. Соответствие деклараций и процедур устанавливает следующая таблица:

процедура:	`center`	`flushleft`	`flushright`
декларация:	`\center`	`\flushleft`	`\flushright`
декларация:	`\centering`	`\raggedright`	`\raggedleft`

Декларации удобно использовать внутри других процедур, соответствующий пример имеется в следующем разделе.

5.2. Выделение абзацев

LaTeX предлагает две процедуры для выделения абзацев:

`\begin{quote}`	...	`\end{quote}`
`\begin{quotation}`	...	`\end{quotation}`

Они форматируют текст в виде колонки, у которой правая и левая границы равно отстоят от правой и левой границ окружающего текста. Процедура `quote` не делает отступ в начале абзаца, а процедура `quotation` делает стандартный отступ.

Процедуру `quote` используют для выделения одного абзаца или коротких цитат:

```
Народная мудрость гласит:
\begin{quote}
Краткость~--- сестра таланта.

Слово~--- серебро, а
молчание~--- золото.
\end{quote}
```

Народная мудрость гласит:

> Краткость — сестра таланта.
>
> Слово — серебро, а молчание — золото.

Процедура `quotation` используется для выделения более одного абзаца:

```
В отличие от процедуры \texttt{quote}
каждый абзац начинается с отступа:
\begin{quotation}
Для коротких фраз используется
\texttt{quote}.

Для выделения двух и более абзацев
лучше подходит \texttt{quotation}.
\end{quotation}
```

В отличие от процедуры `quote` каждый абзац начинается с отступа:

> Для коротких фраз используется `quote`.
>
> Для выделения двух и более абзацев лучше подходит `quotation`.

В теле процедуры `quote` и ей подобных можно использовать декларации `\center`, `\flushleft` и `\flushright`, соответствующие процедурам из предыдущего раздела.

```
Вот пример цитаты с выравниванием:
\begin{quote}\flushright
в процедуре \texttt{quote}
использовано выравнивание текста\\
вправо.
\end{quote}
```

Вот пример цитаты с выравниванием:

в процедуре `quote`
использовано
выравнивание текста
вправо.

5.3. Стихи

Для форматирования стихов используется процедура `verse`:

```
\begin{verse} ... \end{verse}
```

Она устанавливает одинаковый отступ от правой и левой границы. Строки отделяются друг от друга командой `\\`. Строфы, как обычные абзацы, разделяются пустыми строками.

> Глухой глухого звал к суду судьи глухого,
> Глухой кричал: «Моя им сведена корова!» —
> «Помилуй,— возопил глухой в ответ,—
> Сей пустошью владел ещё покойный дед».

Судья решил: «Чтоб не было разврата[1],
Жените молодца, хоть девка виновата».

А. С. Пушкин

Эти шутливые стихи во входном файле выглядят примерно так:

```
\begin{verse}
  Глухой глухого звал к суду судьи глухого,\\*
  Глухой кричал: <<Моя им сведена корова!>> ---\\
  <<Помилуй,--- возопил глухой в ответ,---\\*
  Сей пустошью владел ещё покойный дед>>.

  Судья решил: <<Чтоб не было разврата, \\*
  Жените молодца, хоть девка виновата>>.

 \emph{A.\,C. Пушкин}
\end{verse}
```

Команда `\\*` действует так же, как и `\\`, но к тому же препятствует переносу следующей строки на новую страницу. Она может использоваться, чтобы запретить перенос строк там, где это нежелательно (раздел 4.4). Поскольку строки в данном случае заметно короче ширины страницы, кажется, что отступ справа больше, чем слева. Равенство отступов будет заметно, если не использовать команды `\\` для разбиения строк. Так иногда поступают, когда процедуру `verse` используют для иных целей, нежели печать поэтических сборников.

5.4. Списки

LaTeX предлагает три процедуры для составления списков:

`\begin{itemize}`	*item-list*	`\end{itemize}`
`\begin{enumerate}`	*item-list*	`\end{enumerate}`
`\begin{description}`	*item-list*	`\end{description}`

Тело процедур *item-list* состоит из последовательности записей, начинающихся с команды

⚠ `\item[`*mark*`]`

Команда `\item` помечает запись меткой *mark*. Если необязательный аргумент (вместе с квадратными скобками) отсутствует, используется «метка по умолчанию». Вид «метки по умолчанию» зависит как от используемой процедуры, так и от уровня вложенности списка в другие процедуры составления списков. Допускаются четыре уровня вложенности. Покажем действие каждой процедуры на примерах.

[1] Имеется несколько вариантов этой строки. Мы не знаем, какой из них действительно принадлежит Александру Сергеевичу.

5.4.1. Процедура itemize

В приводимом ниже примере форма исходного текста во входном файле не имеет значения. Поэтому пробелы в начале некоторых строк необязательны, но они позволяют легче отслеживать уровень вложенности процедуры при внесении в неё изменений.

```
\begin{itemize}
\item Первая запись первого уровня.
   \begin{itemize}
   \item Первая запись второго уровня.
      \begin{itemize}
      \item Третий уровень.
         \begin{itemize}
         \item Четвертый уровень.
         \end{itemize}
      \end{itemize}
   \item Вторая запись второго уровня.
   \end{itemize}
\item Вторая запись первого уровня.
\end{itemize}
```

Результат действия процедуры `\itemize` показывает, как помечаются записи «по умолчанию»:

- Первая запись первого уровня.
    - Первая запись второго уровня.
        - ∗ Третий уровень.
            - · Четвертый уровень.
    - Вторая запись второго уровня.
- Вторая запись первого уровня.

Чтобы изменить метку, нужно описать её в необязательном аргументе команды `\item`:

```
\begin{itemize}
  \item[\$]   Первая запись
  \item[\#\#] Вторая запись
\end{itemize}
```

$ Первая запись

Вторая запись

Метки в этом примере прижимаются к правому краю отведённого для них поля. Изменить это правило можно с помощью команды `\hfill` (раздел 4.3). Например: `\item[\$\hfill]`.

Метки, используемые процедурой `itemize` по умолчанию на соответствующем уровне вложенности, производятся командами

`\labelitemi`	`\labelitemii`
`\labelitemiii`	`\labelitemiv`

Переопределив их с помощью `\renewcommand` (раздел 7.1), можно заменить метки сразу у всех записей:

```
\begin{itemize}
\renewcommand{\labelitemi}{\#}
  \item Первая запись
  \item Вторая запись
\end{itemize}
```

Первая запись

Вторая запись

Полезно обратить внимание, что здесь команда `\labelitemi` переопределена внутри процедуры `itemize`. Если её изменить до начала процедуры, то новое определение будет использовано в последующих примерах, а мы к этому в данном случае не стремимся. Область действия вновь определённых команд подчиняется тем же правилам, что и область действия деклараций (раздел 2.5).

5.4.2. Процедура `enumerate`

Процедура `enumerate` нумерует записи. Она также может быть вложена в другие процедуры составления списков. Соответственно четырём уровням вложенности используются четыре счётчика: `enumi`, `enumii`, `enumiii`, `enumiv`. Каждая запись увеличивает счётчик соответствующего уровня на единицу. Однако при наличии необязательного аргумента в команде `\item[`*mark*`]` счётчик не увеличивается. Заменив в самом первом примере из предыдущего раздела все `itemize` на `enumerate`, нетрудно установить, как нумеруются записи всех возможных уровней по умолчанию:

1. Первая запись первого уровня.

    (a) Первая запись второго уровня.

        i. Третий уровень.

            A. Четвертый уровень.

    (b) Вторая запись второго уровня.

2. Вторая запись первого уровня.

«Метки по умолчанию» печатают команды

`\labelenumi`	`\labelenumii`
`\labelenumiii`	`\labelenumiv`

Они используют перечисленные выше счётчики. О счётчиках мы рассказывали в разделе 2.9, тем не менее приведём ещё один несложный пример:

```
\begin{enumerate}
\renewcommand{\theenumi}{\Asbuk{enumi}}
  \item Первая запись
  \item Вторая запись
\end{enumerate}
```

А. Первая запись

Б. Вторая запись

В результате переопределения `\theenumi` команда `\labelenumi` (которая по умолчанию определена как `\theenumi.`) теперь печатает номера первого уровня заглавными русскими буквами. Читатель может удивиться, зачем мы переопределили команду `\theenumi`, а не саму команду `\labelenumi`. Конечно, так тоже можно сделать, однако команда `\theenumi` используется ещё и в качестве `ref`-значения для печати перекрёстных ссылок (раздел 3.7). Поэтому полезно предусмотреть возможность перекрёстного цитирования при помощи тех же обозначений, которые используются для нумерации записей в списке `enumerate`. Изменение формата меток упрощается использованием расширенной версии процедуры `enumerate`, которую вводит одноимённый пакет. Он описан в разделе 5.6.1.

Ссылки на записи в процедуре `enumerate` формируются, как обычно, при помощи команды `\label`. Форма представления ссылок, печатаемых командой `\ref`, включает указание на уровень вложенности записи. Например, ссылка на первую запись (а) второго уровня внутри записи 1. первого уровня будет напечатана как 1а.

5.4.3. Процедура `description`

В процедуре `description` «метка по умолчанию» отсутствует. Поэтому обычно в команде `\item[`*mark*`]` должен присутствовать необязательный аргумент. По умолчанию метка *mark* будет напечатана полужирным шрифтом. Шрифт метки можно изменить, поставив подходящую декларацию из раздела 1.11 непосредственно в аргумент команды `\item`. Из сказанного следует, что в качестве процедуры `description` могут выступать обе рассмотренные выше процедуры составления списков. Чтобы не путать Читателя, начнём с простого примера:

```
\begin{description}
\item[Утка] Водоплавающая птица.
\item[Корова] Парнокопытное животное.
\item[\itshape Тяни-Толкай]
     Сказочное животное.
\end{description}
```

Утка Водоплавающая птица.

Корова Парнокопытное животное.

Тяни-Толкай Сказочное животное.

Если необязательный аргумент у команды `\item` опущен, а следующий отличный от пробела символ в тексте есть `[`, то LaTeX ошибочно примет эту квадратную скобку за начало необязательного аргумента. Чтобы избежать этой редкой ошибки, символ `[` (возможно, вместе с соседними символами) достаточно заключить в фигурные скобки. В следующем примере слово `[аб]` является меткой записи `\item` и поэтому набрано полужирным шрифтом, слово `[вг]` — первым словом

текста и поэтому смещено вправо, а [де] — вновь меткой, но уже с квадратными скобками:

```
\begin{description}
\item[аб] --- метка,
\item{[вг]} --- первое слово,
\item[{[де]}] --- метка со скобками.
\end{description}
```

аб — метка,

[вг] — первое слово,

[де] — метка со скобками.

В последнем пункте закрывающая квадратная скобка] внутри необязательного аргумента заключена в фигурные скобки, чтобы отличить её от квадратной скобки], означающей конец необязательного аргумента. Подобный приём обращения с необязательным аргументом может быть полезен для любой другой команды, имеющей только необязательные аргументы (см. раздел 2.2).

5.4.4. Процедуры `list` и `trivlist`

Выше были описаны простейшие процедуры форматирования списков. Они и многие другие подобные процедуры определены посредством процедур `list` и `trivlist`. Эти две процедуры позволяют легко управлять всеми параметрами списков, например шириной меток, величиной отступа в начале абзаца или шириной правого и левого полей.

Процедура `list` имеет два аргумента *def-lab* и *decls*:

`\begin{list}{`*def-lab*`}{`*decls*`}` *item-list* `\end{list}`

Тело процедуры *item-list* состоит из записей, каждая из которых начинается с команды `\item[`*mark*`]`. Первый аргумент *def-lab* определяет, как помечаются записи по умолчанию, если необязательный аргумент [*mark*] команды `\item` опущен. Второй аргумент *decls* устанавливает декларации, управляющие форматированием записей. Перед выполнением деклараций в *decls* выполняется одна из команд `\@listi`, `\@listii`, `\@listiii`, `\@listiv` в зависимости от того, сколько перед этим было вложений. Эти команды определяются классом печатного документа. Они устанавливают значения деклараций по умолчанию. Явное использование какой-либо декларации в *decls*, таким образом, переопределяет её значение. Все декларации, которые могут появляться в *decls*, перечислены ниже и показаны на рис. 5.1, иллюстрирующем расположение записей на странице. Все вертикальные размеры — растяжимые длины, а горизонтальные — нерастяжимые (раздел 2.10).

`\topsep` — величина вертикального пробела (дополнительно к обычному пробелу между абзацами `\parskip`, см. раздел 17.2), который вставляется между предшествующим текстом и первой записью списка, а также между последней записью и последующим текстом. Значение по умолчанию устанавливается командой `\@list`*cmd* соответствующего уровня, то есть `\@listi`, `\@listii`, `\@listiii` или `\@listiv`.

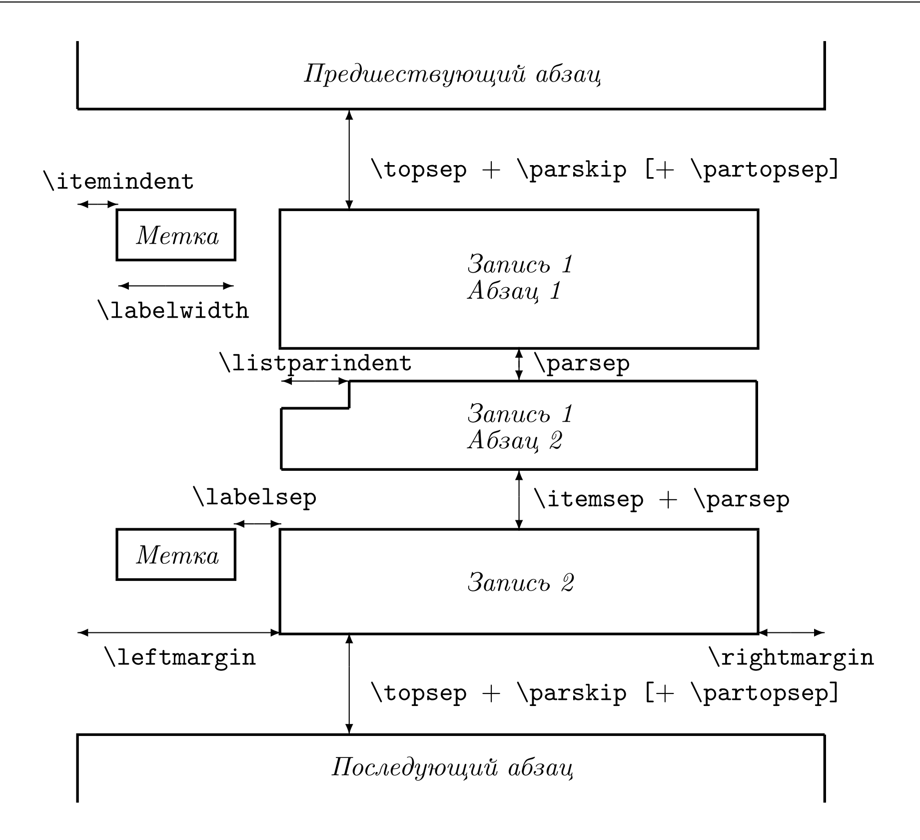


Рис. 5.1. Формат списка

`\partopsep` — дополнительный вертикальный пробел (в дополнение к `\topsep` +`\parskip`), который вставляется между предшествующим текстом и первым пунктом списка, если перед процедурой стоит пустая строка, или между последним пунктом и последующим текстом, если имеется пустая строка после процедуры. Значение по умолчанию устанавливается командой `\@list`*cmd*.

`\parsep` — величина вертикального пробела между абзацами в записи, к которому приравнивается `\parskip` (раздел 17.2) внутри списка. Значение по умолчанию устанавливается командой `\@list`*cmd*.

`\itemsep` — величина дополнительного вертикального пробела (в дополнение к `\parsep`), вставляемая между последовательными записями в списке. Значение по умолчанию устанавливается командой `\@list`*cmd*.

`\leftmargin` — горизонтальное расстояние между левыми границами списка и внешнего текста. Оно должно быть неотрицательным. В стандартных классах печатных документов `\leftmargin` приравнивается к `\leftmargini` командой

`\@listi`, к `\leftmarginii` — командой `\@listii` и т. д.

`\rightmargin` — горизонтальное расстояние между правыми границами списка и внешнего текста. Оно должно быть неотрицательным. Значение по умолчанию равно нулю, если не установлено командой `\@list`*cmd*.

`\listparindent` — величина дополнительного отступа, добавляемая к первой строке каждого абзаца, за исключением первой строки пункта. Может иметь отрицательное значение. Значение по умолчанию равно нулю, если не установлено командой `\@list`*cmd*.

`\itemindent` — величина дополнительного отступа, добавляемая к каждой записи перед меткой. Может иметь отрицательное значение. По умолчанию равна нулю, если не установлена командой `\@list`*cmd*.

`\labelsep` — расстояние между боксом, содержащим метку, и текстом записи. Может иметь отрицательное значение. В печатном документе стандартного класса не изменяется командами `\@list`*cmd*, чтобы обеспечить одно и то же значение для всех уровней вложенности.

`\labelwidth` — ширина бокса, содержащего метку; должна быть неотрицательной. Команда `\@list`*cmd* соответствующего уровня приравнивает её к `\leftmargin`*cmd* − `\labelsep`, так что левый край бокса метки выравнивается по левой границе внешнего текста. Если ширина метки больше, чем `\labelwidth`, то бокс расширяется до ширины метки.

`\makelabel{`*mark*`}` — команда, формирующая метку, которая будет напечатана командой `\item` при наличии у неё необязательного аргумента `[`*mark*`]`. Если команда `\makelabel` не переопределена в `\@list`*cmd*, то метка по умолчанию сдвигается к правому краю бокса. Команда `\makelabel` может быть переопределена командой `\renewcommand`.

В *decls* в дополнение к вышеперечисленным может появиться следующая декларация:

`\usecounter{`*ctr*`}` указывает счётчик *ctr* (глава 7), который будет использован для нумерации записей. Обычно новый счётчик определяется командой `\newcounter`, при этом его значение инициализируется нулем и наращивается командой `\refstepcounter` при выполнении каждой команды `\item`, не имеющей необязательного аргумента. Одновременно значение счётчика назначается текущим `ref`-значением для организации перекрёстного цитирования (раздел 3.7).

В заключение приведём пример, показывающий, как определить список пронумерованных записей.

```
\newcounter{N}
...текст, предшествующий списку.
\begin{list}{\#\arabic{N}}{\usecounter{N}}
\item Это первая запись списка. Обратите
 внимание на правую границу списка.
\item Это вторая запись.
\end{list}
```

…текст, предшествующий списку.

#1 Это первая запись списка. Обратите внимание на правую границу списка.

#2 Это вторая запись.

Процедура

`\begin{trivlist}` *item-list* `\end{trivlist}`

действует подобно процедуре `list`, используя текущие значения деклараций, управляющих списком, за исключением того, что `\parsep` приравнивается к текущему значению `\parskip` (а не наоборот, как в процедуре `list`), а `\leftmargin`, `\labelwidth`, `\itemindent` приравниваются к нулю. Команды `\@list`*cmd* не выполняются.

Каждая команда `\item` в *item-list* должна иметь необязательный аргумент. Процедура `trivlist` обычно используется при создании процедур, состоящих из одной записи, с командой `\item[]`, появляющейся как часть определения процедуры. Например, процедура `center` определена как

```
\begin{trivlist}\centering\item[] ... \end{trivlist}
```

5.5. Неформатированный текст

Процедуры

`\begin{verbatim}`	...	`\end{verbatim}`
`\begin{verbatim*}`	...	`\end{verbatim*}`

печатают текст в точности так, как он записан во входном файле, включая пробелы, специальные символы и команды. При этом команды не исполняются, за исключением `\end{verbatim}` или `\end{verbatim*}`, которые завершают исполнение процедур. В печатном документе соответствующая часть входного файла будет напечатана прямым машинописным шрифтом:

```
\begin{verbatim}
Добро пожаловать в {#}%љ&$_^~!
Выражайтесь яснее, \TeX{}перт!
\end{verbatim}
```

```
Добро пожаловать в {#}%љ&$_^~!
Выражайтесь яснее, \TeX{}перт!
```

Процедура `verbatim*` отличается от `verbatim` тем, что пробел печатается как символ ␣. Процедуры `verbatim` и `verbatim*` не должны появляться в аргументах любых команд, а между `\end` и `{verbatim*}` или `{verbatim}` не должно быть пробела.

Команды

\verb⟨c⟩*text*⟨c⟩
\verb*⟨c⟩*text*⟨c⟩

действуют аналогично процедурам `verbatim` и `verbatim*`, но используются для печати небольших фрагментов текста внутри абзацев. Здесь ⟨c⟩ — любой символ (исключая пробел и звёздочку), который отсутствует в *text*, а *text* — любая последовательность символов, не содержащая ⟨c⟩ (а также невидимого символа конца строки). В печатном документе *text* будет воспроизведен машинописным шрифтом. Как и процедура `verbatim*`, команда `\verb*` печатает пробел символом ␣:

```
Команда \verb*"\ " вставляет
принудительный пробел.
```

Команда \␣ вставляет принудительный пробел.

Команды `\verb` и `\verb*` также не должны появляться в аргументах любых других команд.

5.6. Расширенные процедуры форматирования

Возможности рассмотренных выше процедур форматирования абзацев расширяются при загрузке дополнительных пакетов. Напомним, что соответствующие пакеты должны быть загружены в преамбуле командой `\usepackage`.

5.6.1. Пакет enumerate

Пакет enumerate из коллекции tools переопределяет одноимённую процедуру, упрощая способ изменения формата меток записей, который теперь можно указать в опции процедуры:

\begin{enumerate}[*marks*] *item-list* \end{enumerate}	(enumerate)

Опция *marks* специфицирует формат меток записей. Она может содержать один из символов A, a, I, i или 1, которые отвечают представлению счётчика *ctr*, нумерующего записи, соответственно при помощи команд `\Alph{`*ctr*`}`, `\alph{`*ctr*`}`, `\Roman{`*ctr*`}`, `\roman{`*ctr*`}` или `\arabic{`*ctr*`}`. Более того, *marks* может содержать любые другие символы или команды LaTeX'а, допустимые в необязательном аргументе команды `\item`; однако символы A, a, I, i и 1 должны быть окружены фигурными скобками, если их не следует интерпретировать как представление счётчика. Следующий пример показывает, как оформить процедуру `enumerate` с опцией и создать перекрёстные ссылки на её записи.

```
\begin{enumerate}[{A}I.]
\item Первая запись первого уровня.
\item Вторая запись первого уровня.              \label{La}
   \begin{enumerate}[Абзац a)]
```

```
    \item Первая запись второго уровня.
    \item Вторая запись второго уровня.           \label{Lb}
    \end{enumerate}
    \begin{enumerate}[Абзац a]
    \renewcommand{\alph}[1]{\asbuk{#1}}
    \item Первая запись второго списка.
    \item Вторая запись второго списка.           \label{Lc}
    \end{enumerate}
  \end{enumerate}
  Здесь помечены записи: \ref{La}, \ref{Lb}, \ref{Lc}.
```

И вот что из этого должно получиться:

AI. Первая запись первого уровня.

AII. Вторая запись первого уровня.

Абзац a) Первая запись второго уровня.
Абзац b) Вторая запись второго уровня.

Абзац а) Первая запись второго списка.
Абзац б) Вторая запись второго списка.

Здесь помечены записи II, IIb, IIб.

Существенно, что команда `\ref`, печатающая ссылки, использует только заданное символами A, a, I, i, 1 представление счётчика, не обращая внимание на другие «украшения» в необязательном аргументе процедуры `enumerate`, в том числе на идентичные латинским по начертанию русские буквы А и а, как в слове «Абзац». Обратите внимание, как мы подменили латинские буквы русскими во втором списке, заменив команду `\alph` командой `\asbuk` при помощи `\renewcommand`. Поскольку подмена произведена внутри тела процедуры `enumerate`, сразу вслед за `\begin{enumerate}`, то после `\end{enumerate}` команда `\alph` вновь будет печатать буквы латинского алфавита, как ей и положено.

5.6.2. Пакет alltt

Пакет `alltt`, поставляемый вместе с LaTeX'ом, вводит одноимённую процедуру:

`\begin{alltt} ... \end{alltt}` (alltt)

Она очень похожа на процедуру `verbatim`, но в ней `\`, `{`, `}` сохраняют своё обычное значение. Таким образом, другие команды и процедуры могут появляться внутри процедуры `alltt`. В результате оказывается возможным изменить шрифт внутри процедуры с помощью, например, команды `\emph`, ввести файл при помощи команды `\input` или вставить математическую формулу.

```
\begin{alltt}
В процедуре \emph{alltt} можно
переключать шрифты и набирать
математические формулы. Например:
\( x\sp{2}+y\sp{2}=Z\sb{0}. \)
\end{alltt}
Пробелы учитываются!
```

В процедуре *alltt* можно
переключать шрифты и набирать
математические формулы. Например:
$x^2 + y^2 = Z_0$.

Пробелы учитываются!

Команды `\(`, `\sp`, `\sb` и `\)`, использованные здесь, чтобы набрать формулу $x^2 + y^2 = Z_0$, мы объясним в главе 6. Отметим только, что вне процедуры `alltt` ту же формулу можно было бы записать: `$x^{2}+y^{2}=Z_{0}$`, однако внутри `alltt` символы `$`, `^`, `_` не имеют командного значения, коим они обладают в математических формулах.

5.6.3. Пакет verbatim

Пакет `verbatim` из коллекции tools изменяет одноимённую процедуру так, чтобы избавиться от двух ограничений в её применении. Метод обращения к процедуре прежний:

`\begin{verbatim}`	...	`\end{verbatim}`	(verbatim)
`\begin{verbatim*}`	...	`\end{verbatim*}`	

Во-первых, теперь допускаются пробелы между `\end` и `{verbatim}`. Во-вторых, процедура может печатать фрагменты текста неограниченных размеров, тогда как стандартный метод её реализации, перешедший по наследству от LaTeX 2.09, теоретически может вызвать переполнение памяти компьютера. Платой за новый метод стало одно следствие, заставившее разработчиков LaTeX 2_ε отказаться от полной замены старого метода новым. Теперь текст, следующий за `\end{verbatim}` в той же строке, будет полностью игнорирован. LaTeX предупредит об этом сообщением:

```
LaTeX warning: Characters dropped after \end{verbatim}!
```

Эта проблема легко решается: во входном файле этот текст нужно перенести на следующую строку.

В дополнение к `verbatim` пакет вводит процедуру `comment`, которая пропускает любой текст между `\begin{comment}` и `\end{comment}`:

`\begin{comment} ... \end{comment}` (verbatim)

Кроме того, он определяет команды

`\verbatiminput{`*file*`}`
`\verbatiminput*{`*file*`}` (verbatim)

которые считывают файл *file* и печатают его содержимое так, как это делают соответственно процедуры `verbatim` и `verbatim*`.

5.6.4. Пакет shortvrb

При частом использовании команды `\verb`, как в этой книге, где в изобилии цитируются имена команд LaTeX'а, крепнет желание как-то сократить комбинации типа `\verb|...|`. Пакет `shortvrb` предлагает удобное решение.

Необходимо выбрать какой-нибудь символ ⟨c⟩, который редко встречается в тексте. Это может быть тот символ, который чаще всего используется в команде `\verb` для выделения границ цитируемого текста. Обычно для этих целей подходит `"` или `|`. Если теперь ввести декларацию

`\MakeShortVerb{\⟨c⟩}` (shortvrb)

то в дальнейшем можно использовать ⟨c⟩...⟨c⟩ взамен `\verb`⟨c⟩...⟨c⟩.

```
\MakeShortVerb{\|}
Теперь \verb|\em|
эквивалентно |\em|.
```

Теперь `\em` эквивалентно `\em`.

Декларация

`\DeleteShortVerb{\⟨c⟩}` (shortvrb)

удаляет необычные свойства символа ⟨c⟩. Повторяя `\MakeShortVerb` с разными аргументами, можно несколько символов одновременно приготовить для использования в качестве эквивалента команды `\verb`. Не беда, если случайно среди таких символов два окажутся одинаковыми: их отменит одна декларация `\DeleteShortVerb`. Декларация `\DeleteShortVerb` игнорируется, если её аргумент ⟨c⟩ не представляет собой сокращение команды `\verb`. Обе декларации печатают на экране сообщение, если статус символа ⟨c⟩ изменяется. Обе декларации являются глобальными, то есть область их действия не ограничивается фигурными скобками. «Сокращённая» команда `\verb` не может находиться в аргументах других команд так же, как и сама команда `\verb`, но её можно использовать в процедуре `verbatim` без всяких побочных явлений.

Нешто я да не пойму
При моём-то при уму?
Л. Филатов. Сказ про Федота-стрельца

Глава 6

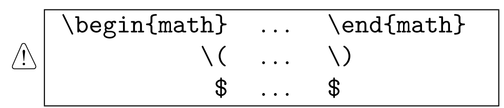
От арифметики до высшей математики

Приверженцы визуального набора математических текстов тратят много сил на конструирование формул, транжиря драгоценное время на выбор подходящего места или шрифта для размещения каждого значка. Это не их дело! С этим прекрасно справится LaTeX! Он воспринимает логическую структуру формул и на этой основе производит их форматирование. Математические выражения на языке LaTeX'а читаются так же, как если бы Читатель проговаривал их вслух, выводя мелом на доске. Впрочем, редакторы, которые способны сохранять визуально набранные формулы в разметке LaTeX, могут быть полезны начинающему пользователю. Можно порекомендовать изделия корпорации Design Science, Inc.[1] Она производит коммерческий редактор формул MathType и его облегченную версию TeXaide, которая распространяется бесплатно. Нужно только помнить, что эти редакторы реализуют лишь малую толику тех возможностей, которые LaTeX предоставляет опытному пользователю.

6.1. Основные процедуры

LaTeX располагает тремя процедурами для форматирования математических формул: `math`, `displaymath` и `equation`, которые включают специальный математический режим форматирования. Его особенности мы рассмотрим чуть позже, а здесь объясним назначение каждой из перечисленных процедур.

Процедура `math` размещает небольшие формулы, такие как $E = mc^2$, внутри абзаца. Ввиду совершенно особого статуса математики в LaTeX'е есть целых три варианта обращения к этой процедуре:



```
\begin{math}  ...  \end{math}
          \(  ...  \)
           $  ...  $
```

Последний вариант, когда знаки `$` метят начало и конец формулы в исходном тексте, обычно используется только для самых коротких формул. Его преиму-

[1] Адрес в интернете: `http://www.mathtype.com/`.

щество состоит в краткости. Однако в больших формулах легко потерять один из знаков $. Так как оба знака в начале и конце формулы одинаковы, в результате потери LaTeX принимает за математическую формулу совсем не то, что следует. Первые два варианта не имеют такого недостатка.

Процедура displaymath создаёт так называемое выключное уравнение, размещая его в отдельной строке:

```
...в отдельной строке:
\[
    \sum_{n=1}^{\infty}
    \frac{(-1)^n}{1+n}=\ln2.
\]
```

...в отдельной строке:

$$\sum_{n=1}^{\infty}\frac{(-1)^n}{1+n}=\ln 2.$$

Обращение к процедуре displaymath имеет два варианта:

```
\begin{displaymath}  ...  \end{displaymath}
                 \[  ...  \]
```

⚠

TeX имеет собственный аналог процедуры displaymath, когда выключная формула записывается в виде $$...$$. Строго говоря, третий вариант процедуры math также есть прямое обращение к TeX'у, минуя надстройку LaTeX. Однако, в отличие от $...$, использование $$...$$ не является полным аналогом процедуры LaTeX'а. Такие мелочи иногда несущественны, но один случай, когда выключные формулы, записанные в виде $$...$$, не подчиняются правилам LaTeX'а, будет отмечен в конце этого раздела.

Процедура

```
\begin{equation}  ...  \end{equation}
```

производит выключные уравнения и автоматически нумерует их. Пометив уравнение с помощью команды \label, можно организовать ссылку на него в любом месте печатного документа.

```
\begin{equation}
    \label{math/1}
    \int_{-\infty}^{\infty}
     e^{-x^2}\,dx=\sqrt{\pi}.
\end{equation}
Это уравнение~(\ref{math/1}) на
стр.~\pageref{math/1}.
```

$$\int_{-\infty}^{\infty} e^{-x^2}\,dx=\sqrt{\pi}. \qquad (6.1)$$

Это уравнение (6.1) на стр. 130.

Все перечисленные процедуры предназначены для сравнительно коротких формул, умещающихся в одной строке. LaTeX умеет переносить со строки на строку формулы, формируемые процедурой math. Обычно разрыв производится на знаке равенства[2]. Однако это не решает проблему размещения очень больших формул или систем уравнений. Поэтому LaTeX имеет ещё процедуру array

[2] Место разрыва может быть установлено командой \discretionary (раздел 4.4) при условии, что не загружены пакеты AMS-LaTeX (глава 8).

для создания матриц и процедуру `eqnarray` для форматирования многострочных формул или систем уравнений. Эти процедуры описаны в разделах 6.4.5 и 6.8.

Процедура `math` может использоваться в текстовой и строковой моде, процедура `array` — в математической, а остальные процедуры — только в текстовой моде.

Все выключные формулы выравниваются по центру строк, если в команде `\documentclass` не указана опция `fleqn`. Эта опция устанавливает, что выключные формулы располагаются на заданном расстоянии от левой границы окружающего текста. Однако опция `fleqn` не действует на выключные формулы, созданные при помощи `$$...$$`, поэтому запись вида `$$...$$` следует избегать.

Если выключная формула имеет номер, то по умолчанию он располагается справа. Опция `leqno` в `\documentclass` изменяет правую нумерацию формул на левую.

Все процедуры, нумерующие формулы, используют один и тот же счётчик `equation`, что обеспечивает единую нумерацию формул. Формат номера формулы определяется классом печатного документа. Например, в классах `book` и `report` формулы нумеруются независимо в пределах каждой главы (`\chapter`), а номер формулы содержит также указание на номер главы. Формат номера может быть изменён посредством переопределения команды `\theequation` в соответствии с общими правилами работы со счётчиками (раздел 2.9).

В главе 8 описаны пакеты $\mathcal{AMS}$-LaTeX. Они вводят ещё несколько процедур для форматирования систем уравнений.

6.2. От простого к сложному

Самая простая формула состоит из одной буквы x, которая в исходном тексте записывается в виде `$x$`. На первый взгляд, она ничем не отличается от курсивного текста `\textit{x}`, но уже следующий пример показывает, что в математической моде LaTeX использует иные правила кернинга, чем в обычном тексте.

```
Сравните: \textit{ffi} и $ffi$.
```

Сравните: *ffi* и ffi.

Символ `-`, известный нам в роли дефиса (-), превращается в знак минус ($-$); символы `<` и `>`, которые в обычном тексте отображаются по-разному в зависимости от текущего языка, превращаются в знаки сравнения ($<$ и $>$); символ `*` в тексте и формуле позиционируется разными способами: (*) и ($*$) соответственно.

В отличие от букв, цифры в математических формулах печатаются прямым шрифтом.

```
Сравните: 1+2=3 и $1+2=3$.
```

Сравните: 1+2=3 и $1+2=3$.

Математики любят «украшать» свои формулы греческими буквами и другими экзотическими значками. На сухом языке формул фраза «LaTeX великолепен» может выглядеть так:

```
$ \Lambda\alpha\tau\epsilon\chi \to \infty $
```

$\Lambda\alpha\tau\epsilon\chi \to \infty$

Специальные математические символы производятся командами, которые перечислены в таблицах 6.1–6.13. Имена команд, часто очень длинные, совпадают с названиями символов. Это удобно, поскольку такие имена «всегда на слуху». Например, греческая буква β (бета) так и пишется: `\beta`. Неудобства, вызванные длинным написанием некоторых команд, весьма относительны, так как для часто используемых команд всегда можно ввести краткие синонимы (глава 7).

В первом приближении набор специальных математических символов очень прост: отыскав в таблице команду, соответствующую требуемому символу, достаточно ввести её в исходный текст печатного документа. Специализированные редакторы, такие как WinEdt или TeXnicCenter, имеют разветвлённое меню, где есть наиболее употребительные математические символы. Однако столь простой подход не позволяет реализовать всю мощь LaTeX'а. Например, прописные греческие буквы, как Ω, обычно набираются прямым шрифтом, а строчные, как ψ,— курсивом. А если нужна курсивная $\mathit{\Omega}$, полужирная $\boldsymbol{\psi}$ или полужирный знак квадратного корня $\sqrt{\ }$? В математических формулах нельзя[3] использовать декларации, изменяющие гарнитуру шрифта в обычном тексте (`\rmfamily`, `\itshape` и т. д.), а также соответствующие им команды (`\textrm`, `\textit`). Вместо них имеются команды, предназначенные специально для математических формул. Более того, нельзя менять размер шрифта внутри математических формул. Вопросы, связанные с математическими шрифтами, мы обсудим подробнее в разделах 6.5 и 6.6. Читатель может пропустить эти разделы при первом чтении, так как выбор шрифтов, предлагаемый LaTeX'ом по умолчанию, удовлетворяет большинству потребностей.

LaTeX игнорирует все пробелы в исходном тексте математических формул. Например, записи `$ x$` или `$ 2$` означают то же самое, что и `$x$` или `$2$`. Читатель может набрать `$(x - y)/(x + y)$` или `$(x-y)/(x+y)$`, но в любом случае получит одну и ту же формулу $(x - y)/(x + y)$, в которой имеются небольшие промежутки вокруг знаков $+$ и $-$, но не вокруг знака $/$. Таким образом, нет нужды запоминать сложные правила кернинга математических выражений: пробелы можно ставить так, как удобно. Однако они по-прежнему играют роль признака окончания имени команды, как об этом рассказано в разделе 2.1. Кроме того, исходный текст математических формул не должен содержать пустых строк. В большинстве случаев правила кернинга математических формул, которые использует LaTeX, будут восприняты любым математиком как вполне естественные. При необходимости они могут быть изменены при помощи команд, описанных в разделе 6.7.

LaTeX компонует сложные формулы из менее сложных математических выражений простым и логичным способом так же, как это делает школьник, изучивший простейшие правила алгебры. LaTeX сам выберет размер скобок, дробной черты, знака корня, расстояние между символами и знаками математических операций. Искушённый Читатель должен обратиться к главе 8, если ему нужно изменить толщину дробной черты или сделать нечто ещё более редкое. Мы же

[3] Нельзя в том смысле, что результат не гарантирован.

будем последовательны и начнём с самого главного — с алфавита математики.

6.3. Алфавит математики

LaTeX имеет в своём багаже сотни специальных математических символов. Лишь мизерная толика из них имеется на клавиатуре компьютеров. Отсутствующие символы производятся командами, собранными в таблицах 6.1–6.13. Эти команды могут применяться только в математической моде.

Математические символы подразделяются на несколько классов, различающихся, главным образом, по способу позиционирования символов относительно соседних. Символы каждого класса сгруппированы в отдельной таблице. Если какой-либо символ присутствует в двух классах, он обычно производится разными командами. Если какая-либо команда присутствует в двух таблицах, её исполнение зависит от контекста, в котором она использована. Следующие ниже комментарии к таблицам ознакомят Читателя с такими особенностями.

6.3.1. Символы с диакритическими знаками

Команды расстановки диакритических знаков, рассмотренные в разделе 4.1, не работают в математической моде, но их можно использовать внутри команды `\mbox`:

```
$a+\mbox{\u{b}}=\mbox{\c{c}}$
```

| $a + \breve{\mathrm{b}} = \text{ç}$

Однако для всех диакритических знаков, обычно встречающихся в математических выражениях, имеются специальные команды, действующие только в математической моде. Они перечислены в таблице 6.1.

```
$\vec{r}=(\hat{x},\acute{y},\tilde{z})$
```

| $\vec{r} = (\hat{x}, \acute{y}, \tilde{z})$

Команды

```
\widehat{math}
\widetilde{math}
```

производят широкие версии наиболее употребительных диакритических знаков `\hat` и `\tilde`.

```
$\widehat{x}$, $\widehat{-x}$, $\widehat{x-y}$
```

| $\widehat{x}$, $\widehat{-x}$, $\widehat{x-y}$

Широкие диакритические знаки могут «покрыть» до трёх символов.

6.3.2. Греческие буквы

Имена команд, соответствующих греческим буквам, получаются присоединением обратного слеша \ к названию буквы в английской транскрипции (таблица 6.2), причём прописным греческим буквам соответствуют команды, начинающиеся с

Таблица 6.1

Математические диакритические знаки

$\hat{x}$	\hat{x}	$\acute{x}$	\acute{x}	$\bar{x}$	\bar{x}	$\dot{x}$	\dot{x}
$\check{x}$	\check{x}	$\grave{x}$	\grave{x}	$\vec{x}$	\vec{x}	$\ddot{x}$	\ddot{x}
$\breve{x}$	\breve{x}	$\tilde{x}$	\tilde{x}	$\mathring{x}$	\mathring{x}		

Таблица 6.2

Строчные греческие буквы

α	\alpha	θ	\theta	τ	\tau	o	o
β	\beta	ϑ	\vartheta	υ	\upsilon	π	\pi
γ	\gamma	ι	\iota	ϖ	\varpi	ϕ	\phi
δ	\delta	κ	\kappa	ρ	\rho	φ	\varphi
ϵ	\epsilon	λ	\lambda	ϱ	\varrho	χ	\chi
ε	\varepsilon	μ	\mu	σ	\sigma	ψ	\psi
ζ	\zeta	ν	\nu	ς	\varsigma	ω	\omega
η	\eta	ξ	\xi				

Таблица 6.3

Прописные греческие буквы

Γ	\Gamma	Λ	\Lambda	Σ	\Sigma	Ψ	\Psi
Δ	\Delta	Ξ	\Xi	Υ	\Upsilon	Ω	\Omega
Θ	\Theta	Π	\Pi	Φ	\Phi		

Таблица 6.4

Дополнительные символы

<table>
<tr><td>∞</td><td>\infty</td><td>$\prime$</td><td>\prime</td><td>$\forall$</td><td>\forall</td><td>$\aleph$</td><td>\aleph</td></tr>
<tr><td>$\Box$</td><td>\Box†</td><td>$\emptyset$</td><td>\emptyset</td><td>$\exists$</td><td>\exists</td><td>$\hbar$</td><td>\hbar</td></tr>
<tr><td>$\Diamond$</td><td>\Diamond†</td><td>∇</td><td>\nabla</td><td>$\neg$</td><td>\neg</td><td>$\imath$</td><td>\imath</td></tr>
<tr><td>$\triangle$</td><td>\triangle</td><td>$\surd$</td><td>\surd</td><td>$\flat$</td><td>\flat</td><td>$\jmath$</td><td>\jmath</td></tr>
<tr><td>$\clubsuit$</td><td>\clubsuit</td><td>$\top$</td><td>\top</td><td>$\natural$</td><td>\natural</td><td>ℓ</td><td>\ell</td></tr>
<tr><td>$\diamondsuit$</td><td>\diamondsuit</td><td>$\bot$</td><td>\bot</td><td>$\sharp$</td><td>\sharp</td><td>$\wp$</td><td>\wp</td></tr>
<tr><td>$\heartsuit$</td><td>\heartsuit</td><td>$\|$</td><td>\|</td><td>$|$</td><td>|</td><td>$\Re$</td><td>\Re</td></tr>
<tr><td>$\spadesuit$</td><td>\spadesuit</td><td>$\angle$</td><td>\angle</td><td>∂</td><td>\partial</td><td>$\Im$</td><td>\Im</td></tr>
<tr><td>$\mho$</td><td>\mho†</td><td>$\smallint$</td><td>\smallint</td><td>$\backslash$</td><td>\backslash</td><td></td><td></td></tr>
</table>

(помеченные команды† определены в пакете latexsym)

Таблица 6.5

Символы бинарных операций

$+$	`+`	$-$	`-`	$\diamond$	`\diamond`	$\pm$	`\pm`
$\oplus$	`\oplus`	$\cap$	`\cap`	$\bigtriangleup$	`\bigtriangleup`	$\mp$	`\mp`
$\ominus$	`\ominus`	$\cup$	`\cup`	$\bigtriangledown$	`\bigtriangledown`	$\times$	`\times`
$\otimes$	`\otimes`	$\uplus$	`\uplus`	$\triangleleft$	`\triangleleft`	$\div$	`\div`
$\oslash$	`\oslash`	$\sqcap$	`\sqcap`	$\triangleright$	`\triangleright`	$\ast$	`\ast`
$\odot$	`\odot`	$\sqcup$	`\sqcup`	$\lhd$	`\lhd`†	$\star$	`\star`
$\bigcirc$	`\bigcirc`	$\vee$	`\vee`	$\rhd$	`\rhd`†	$\circ$	`\circ`
$\dagger$	`\dagger`	$\wedge$	`\wedge`	$\unlhd$	`\unlhd`†	$\bullet$	`\bullet`
$\ddagger$	`\ddagger`	$\setminus$	`\setminus`	$\unrhd$	`\unrhd`†	$\cdot$	`\cdot`
$\amalg$	`\amalg`	$\wr$	`\wr`				

(помеченные команды† определены в пакете latexsym)

Таблица 6.6

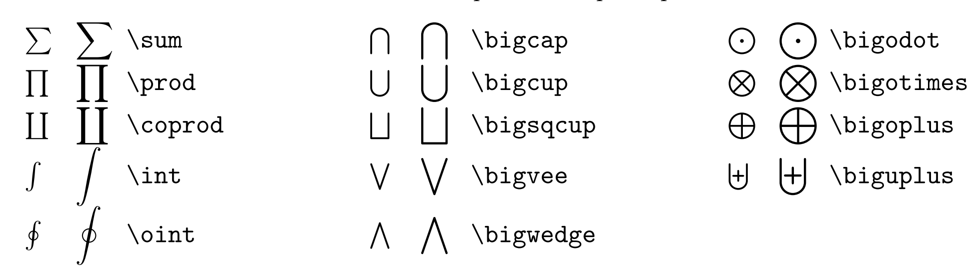
Символы сравнения

$<$	`<`	$>$	`>`	$=$	`=`	$\equiv$	`\equiv`
$\leq$	`\leq`	$\geq$	`\geq`	$\sim$	`\sim`	$\models$	`\models`
$\prec$	`\prec`	$\succ$	`\succ`	$\simeq$	`\simeq`	$\perp$	`\perp`
$\preceq$	`\preceq`	$\succeq$	`\succeq`	$\asymp$	`\asymp`	$\mid$	`\mid`
$\ll$	`\ll`	$\gg$	`\gg`	$\approx$	`\approx`	$\parallel$	`\parallel`
$\subset$	`\subset`	$\supset$	`\supset`	$\cong$	`\cong`	$\neq$	`\neq`
$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\bowtie$	`\bowtie`	$\Join$	`\Join`†
$\sqsubset$	`\sqsubset`†	$\sqsupset$	`\sqsupset`†	$\doteq$	`\doteq`	$\smile$	`\smile`
$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`	$\propto$	`\propto`	$\frown$	`\frown`
$\in$	`\in`	$\ni$	`\ni`	$\vdash$	`\vdash`	$\dashv$	`\dashv`
$:$	`:`	$\notin$	`\notin`				

(помеченные команды† определены в пакете latexsym)

Таблица 6.7

Символы переменного размера



$\sum$	$\displaystyle\sum$	`\sum`	$\bigcap$	$\displaystyle\bigcap$	`\bigcap`	$\bigodot$	$\displaystyle\bigodot$	`\bigodot`
$\prod$	$\displaystyle\prod$	`\prod`	$\bigcup$	$\displaystyle\bigcup$	`\bigcup`	$\bigotimes$	$\displaystyle\bigotimes$	`\bigotimes`
$\coprod$	$\displaystyle\coprod$	`\coprod`	$\bigsqcup$	$\displaystyle\bigsqcup$	`\bigsqcup`	$\bigoplus$	$\displaystyle\bigoplus$	`\bigoplus`
$\int$	$\displaystyle\int$	`\int`	$\bigvee$	$\displaystyle\bigvee$	`\bigvee`	$\biguplus$	$\displaystyle\biguplus$	`\biguplus`
$\oint$	$\displaystyle\oint$	`\oint`	$\bigwedge$	$\displaystyle\bigwedge$	`\bigwedge`			

Таблица 6.8

Разделители

$($	`(`	$)$	`)`	$\uparrow$	`\uparrow`
$[$	`[`	$]$	`]`	$\downarrow$	`\downarrow`
$\{$	`\{`	$\}$	`\}`	$\updownarrow$	`\updownarrow`
$\lfloor$	`\lfloor`	$\rfloor$	`\rfloor`	$\Uparrow$	`\Uparrow`
$\lceil$	`\lceil`	$\rceil$	`\rceil`	$\Downarrow$	`\Downarrow`
$\langle$	`\langle`	$\rangle$	`\rangle`	$\Updownarrow$	`\Updownarrow`
$/$	`/`	$\backslash$	`\backslash`		
$\vert$	`|`	$\Vert$	`\\|`		

Таблица 6.9

Стрелки

$\leftarrow$	`\leftarrow`	$\longleftarrow$	`\longleftarrow`
$\Leftarrow$	`\Leftarrow`	$\Longleftarrow$	`\Longleftarrow`
$\rightarrow$	`\rightarrow`	$\longrightarrow$	`\longrightarrow`
$\Rightarrow$	`\Rightarrow`	$\Longrightarrow$	`\Longrightarrow`
$\leftrightarrow$	`\leftrightarrow`	$\longleftrightarrow$	`\longleftrightarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\Longleftrightarrow$	`\Longleftrightarrow`
$\mapsto$	`\mapsto`	$\longmapsto$	`\longmapsto`
$\hookleftarrow$	`\hookleftarrow`	$\hookrightarrow$	`\hookrightarrow`
$\leftharpoonup$	`\leftharpoonup`	$\rightharpoonup$	`\rightharpoonup`
$\leftharpoondown$	`\leftharpoondown`	$\rightharpoondown$	`\rightharpoondown`
$\rightleftharpoons$	`\rightleftharpoons`	$\leadsto$	`\leadsto`†
$\uparrow$	`\uparrow`	$\Updownarrow$	`\Updownarrow`
$\Uparrow$	`\Uparrow`	$\nearrow$	`\nearrow`
$\downarrow$	`\downarrow`	$\searrow$	`\searrow`
$\Downarrow$	`\Downarrow`	$\swarrow$	`\swarrow`
$\updownarrow$	`\updownarrow`	$\nwarrow$	`\nwarrow`

(помеченные команды† определены в пакете latexsym)

Таблица 6.10

Символы пунктуации

$,$	`,`	$;$	`;`	$\colon$	`\colon`	$.$	`\ldotp`	$\cdot$	`\cdotp`

Таблица 6.11

Многоточия

$\ldots$	`\ldots`	$\cdots$	`\cdots`	$\vdots$	`\vdots`	$\ddots$	`\ddots`

Таблица 6.12

Стандартные функции

\arccos	\arcctg†	\arcsin	\arctan	\arctg†	\arg	\ch†
\cos	\cosh	\cot	\coth	\cosec†	\csc	\ctg†
\cth†	\deg	\det	\dim	\exp	\gcd	\hom
\inf	\ker	\lg	\lim	\liminf	\limsup	\ln
\log	\max	\min	\Pr	\sec	\sh†	\sin
\sinh	\sup	\tan	\tanh	\th†	\tg†	

(помеченные команды† определены в пакете babel с опциями bulgarian, russian, ukrainian.)

Таблица 6.13

Синонимы

$\leq$	\le	(для \leq)	$\geq$	\ge	(для \geq)
$\{$	\lbrace	(для \{)	$\}$	\rbrace	(для \})
$\to$	\to	(для \rightarrow)	$\gets$	\gets	(для \leftarrow)
$\ni$	\owns	(для \ni)	$\neg$	\lnot	(для \wedge)
$\vee$	\lor	(для \vee)	$\wedge$	\land	(для \neg)
$\vert$	\vert	(для |)	$\Vert$	\Vert	(для \\|)
$\neq$	\ne	(для \neq)			

прописной буквы (таблица 6.3). Строчная буква «омикрон» не имеет своей команды, так как совпадает с буквой «o». По той же причине отсутствуют специальные команды для некоторых прописных греческих букв. Полезно также заметить, что буква \upsilon (υ) немного шире, чем v (v); обе эти буквы следует отличать от \nu (ν). Некоторые строчные греческие буквы присутствуют в двух вариантах. Так ε (ε) не нужно путать с ϵ (ϵ), а ϑ (ϑ) с θ (θ).

По умолчанию прописные греческие буквы печатаются прямым шрифтом, а строчные — курсивным. Как изменить это соглашение, рассказано в разделе 6.6.

6.3.3. Дополнительные символы

Обзор специальных математических символов, которые LaTeX позиционирует по правилам, принятым для букв и цифр, завершает таблица 6.4. Буквы \imath ($\imath$) и \jmath ($\jmath$) без точек наверху следует использовать, когда i и j имеют диакритический знак; например, $\hat{\imath}$ производит '$\hat{\imath}$'. Символ \angle ($\angle$) построен из нескольких других, поэтому он не уменьшается, когда появляется в индексах; \partial (∂) обозначает частную производную; \hbar ($\hbar$) называется постоянной Планка, а \| ($\|$) и \bot ($\bot$) используются в нижних индексах для обозначения параллельной и перпендикулярной компонент векторов.

Редко применяемые команды \Box ($\Box$), \Diamond ($\Diamond$), \mho ($\mho$) не определены в формате LaTeX 2$_\varepsilon$ (в отличие от LaTeX2.09 или Plain TeX). Они помечены в табл. 6.4 значком † и доступны после загрузки пакета latexsym.

6.3.4. Бинарные операции и символы сравнения

LaTeX оставляет небольшие промежутки вокруг символов бинарных операций (таблица 6.5), если эти символы стоят между какими-нибудь другими символами. Поэтому `$X \setminus Y$` ($X \setminus Y$) — не то же самое, что и `$X \backslash Y$` ($X\backslash Y$), хотя обе команды \setminus (таблица 6.5) и \backslash (таблица 6.4) используют один и тот же символ. Символ *, имеющийся на клавиатуре, не относится к символам бинарных операций, а правильные пробелы вокруг $\ast$ производит команда \ast ($\ast$).

Команды \mid, \parallel и \perp производят знаки сравнения (таблица 6.6), использующие символы $|$, $\|$ и $\bot$, которые можно также получить, набрав |, \| и \bot (таблицы 6.4 и 6.8). LaTeX окружает символы сравнения (как и символы бинарных операций) небольшими пробелами, когда они действительно используются для сравнения.

```
Сравните: $\vec{A} \| \vec{B}$ и
$\vec{A} \parallel \vec{B}$.
```

Сравните: $\vec{A}\|\vec{B}$ и $\vec{A} \parallel \vec{B}$.

Однако промежутки вокруг символов бинарных операций и символов сравнения исчезают, когда те стоят в индексах.

Команда

```
\not
```

поставленная перед символом бинарной операции или символом сравнения, производит «отрицательный» оператор, перечёркивая его косой чертой:

```
Если $x\not=y$, то \( y\not\leq x+z \).
```

Если $x \not= y$, то $y \not\leq x + z$.

Команда \not перечёркивает любой символ, следующий за ней, хотя позиционирование черты может не быть идеальным.

```
$\not\alpha$, $\not\cup$.
```

$\not\alpha$, $\not\cup$.

Такое положение черты можно поправить, используя команды из раздела 6.7.

6.3.5. Символы переменного размера

Символы из таблицы 6.7 изменяют свой размер в зависимости от того, используются они в формуле внутри абзаца или в выключной формуле в отдельной строке. Во втором случае размер этих символов увеличивается. Большие символы \sum ($\sum$) и \prod ($\prod$) полезно отличать от греческих букв \Sigma (Σ) и \Pi (Π), символ \coprod ($\coprod$) — от символа бинарной операции \amalg ($\amalg$), а \int

($\int$) — от `\smallint` ($\smallint$). Символы, начинающиеся с префикса `\big`, имеют аналоги меньшего размера среди символов бинарных операций. Символы бо́льшего размера обычно используются в начале формулы и имеют индексы в качестве пределов:

```
$\bigcup_{n=1}^{m}(x_{n}\cup y_{n})$
```

$\bigcup_{n=1}^{m}(x_{n}\cup y_{n})$

Позиционирование индексов у некоторых символов переменного размера зависит от того, использован этот символ в формуле внутри абзаца или в выключной формуле. Если формулу из последнего примера поместить в отдельной строке, то верхние и нижние индексы будут размещены соответственно над и под изображением символа.

```
\[ \bigcup_{n=1}^{m}(x_{n}\cup y_{n}) \]
```

$$\bigcup_{n=1}^{m}(x_{n}\cup y_{n})$$

В разделе 6.4.1 рассказано, как изменить это правило.

6.3.6. Разделители

Разделитель — это символ, который, подобно скобкам (и), служит для логического выделения математических выражений. В таблице 6.8 перечислены все символы, которые LaTeX рассматривает в качестве разделителей. Размеры разделителей также могут автоматически подстраиваться под размер формулы. Однако команды из таблицы 6.8 производят разделители фиксированного размера. Чтобы получить разделители бо́льшего размера, соответствующего высоте выражения, нужно перед разделителями, окружающими выражение, поставить команды `\left` и `\right`:

`\left`*delim*$_1$ `...` `\right`*delim*$_2$

Команды `\left` и `\right` должны всегда появляться парами, но сами разделители *delim*$_1$ и *delim*$_2$ могут быть разными, как в следующем примере, где $\lfloor$ и $\rfloor$ окружают дробь, созданную командой `\frac`:

```
\[ \pi(n)=\sum_{k=2}^{n}
   \left\lfloor
        \frac{\phi(k)}{k-1}
   \right\rfloor \]
```

$$\pi(n)=\sum_{k=2}^{n}\left\lfloor\frac{\phi(k)}{k-1}\right\rfloor$$

Если один из разделителей не нужен, его достаточно заменить точкой: команды `\left.` и `\right.` производят невидимые разделители.

```
\[ \frac{a+1}{b} \left/
   \frac{c+1}{d} \right. \]
```

$$\frac{a+1}{b} \left/ \frac{c+1}{d} \right.$$

Вооружившись лупой, дотошный Читатель мог бы заметить, что в последнем примере наклонная дробная черта смещена вправо в сторону знаменателя. Существуют также примеры, когда выбор размера разделителей, осуществляемый

командами \left и \right, не удовлетворяет высоким стандартам LaTeX'а. Например, в формуле $\big||x|+|y|\big|$ внешние символы | желательно сделать больше, чем внутренние, которые окружают буквы x и y. Однако `$\left||x|+|y|\right|$` производит всего лишь $||x| + |y||$. На этот и другие подобные случаи LaTeX имеет серию команд

\bigl	\bigm	\bigr	\big
\Bigl	\Bigm	\Bigr	\Big
\biggl	\biggm	\biggr	\bigg
\Biggl	\Biggm	\Biggr	\Bigg

Команды в первой, второй и третьей колонках предназначены соответственно для разделителей, стоящих слева (l), посередине (m) и справа (r) от математического выражения. LaTeX немного по-разному устанавливает промежутки вокруг правых, средних и левых разделителей. Поэтому наиболее правильный вариант набора формулы $\bigl||x| + |y|\bigr|$ будет таким: `$\bigl||x|+|y|\bigr|$`. Команды в последней колонке создают большие разделители, которые LaTeX позиционирует, как обычные символы. Они используются для увеличения знака деления, как в следующем примере:

```
         Сравните:
\[
  \frac{a+1}{b}\biggm/\frac{c+1}{d}
\]
              и
\[
  \frac{a+1}{b}\bigg /\frac{c+1}{d}.
\]
```

Сравните:

$$\frac{a+1}{b}\biggm/\frac{c+1}{d}$$

и

$$\frac{a+1}{b}\bigg /\frac{c+1}{d}.$$

Команды в серии \big*cmd* увеличивают высоту следующего за ними разделителя на 50% по сравнению с исходной; команды в серии \Big*cmd* — ещё на такую же величину и так далее. Так что максимальный размер разделителей с \Bigg*cmd* больше исходного в 3 раза.

Размер разделителя	( )	\left(\right)	\bigl(\bigr)	\Bigl(\Bigr)	\biggl(\biggr)	\Biggl(\Biggr)
Результат	$(b)(\frac{a}{b})$	$(b)\left(\frac{a}{b}\right)$	$\bigl(b\bigr)\bigl(\frac{a}{b}\bigr)$	$\Bigl(b\Bigr)\Bigl(\frac{a}{b}\Bigr)$	$\biggl(b\biggr)\biggl(\frac{a}{b}\biggr)$	$\Biggl(b\Biggr)\Biggl(\frac{a}{b}\Biggr)$

Дополнительные примеры использования больших разделителей Читатель найдёт в разделах 6.4.5, 6.8 и 12.3.5.

6.3.7. Стрелки

Имеется ещё один класс символов сравнения в виде разнообразных стрелок, представленных в таблице 6.9. Стрелки вверх и вниз увеличиваются в высоту, когда используются в качестве разделителей.

6.3.8. Знаки пунктуации и многоточия

LaTeX вставляет небольшой пробел после знаков пунктуации (таблица 6.10), но не перед ними. В этом смысле точка или восклицательный знак не являются знаками пунктуации в математических выражениях. Однако, если точка набрана посредством команды `\ldotp`, после неё автоматически будет добавлен пробел правильной величины.

`Сравните: $1.1$ и $1\ldotp 1$.`	Сравните: 1.1 и $1\ldotp 1$.

Примеры регулировки пробелов вокруг знаков пунктуации имеются в разделе 6.7.

Существуют многоточия четырёх видов (таблица 6.11). Команда `\ldots` ($\ldots$) работает в любой моде, а `\cdots` ($\cdots$), `\vdots` ($\vdots$) и `\ddots` ($\ddots$) — только в математической. Нижнее горизонтальное многоточие `\ldots` предназначено для выражений типа $a_1, a_2 \ldots a_n$. Центрированное многоточие `\cdots` используют обычно между знаками бинарных операций $+$, $-$ или $=$: $a_1 + \cdots + a_n$.

6.3.9. Стандартные функции

В формуле $\sin(x)$ обозначение функции sin принято печатать прямыми буквами, а не курсивом. Однако простой набор `sin` в исходном тексте обозначает произведение трёх величин: s, i и n. Обозначение функции sin печатает команда `\sin` из таблицы 6.12, где также приведены команды для обозначений других широко используемых функций. Таким образом, функция $\sin(x)$ на языке LaTeX'а записывается в виде `$\sin(x)$`. Некоторые функции по-разному обозначаются в русской и зарубежной литературе. Поэтому пакет babel с опцией `russian` добавляет ещё несколько команд:

`$\tan\to\tg$, $\cot\to\ctg$,\\`	tan → tg, cot → ctg,
`$\cosh\to\ch$, $\sinh\to\sh$,\\`	cosh → ch, sinh → sh,
`$\tanh\to\th$, $\coth\to\cth$,\\`	tanh → th, coth → cth,
`$\csc\to\cosec$, $\arctan\to\arctg$,\\`	csc → cosec, arctan → arctg,
`$\arctan^{-1}\to\arcctg$.`	$\arctan^{-1}$ → arcctg.

Ещё две команды печатают обозначение функции, выделяющей остаток деления на заданное число:

```
\bmod
\pmod{math}
```

причём `$\bmod x$` печатает «mod$x$», а `$\pmod{x}$` — « (mod x)». Индексы у некоторых команд функций позиционируются так же, как у символов переменного размера, то есть в выключных формулах индексы располагаются не сбоку, а над или под обозначением функции (разделы 6.3.5 и 6.4.1).

Если представленный список функций недостаточен, его можно расширить при помощи команды `\DeclareMathOperator`, описанной в разделе 8.11.14.

6.3.10. Синонимы

Некоторые символы имеют альтернативные имена, которые широко используются в математических публикациях. Они представлены в таблице 6.13. Имеется также команда `\iff` ($\iff$), похожая на `\Longleftarrow`, но она добавляет небольшие промежутки вокруг символа. В главе 7 рассказано, как вводить синонимы для любых других команд.

Синонимы `\lbrace` и `\rbrace` для команд `\{` и `\}` были введены, поскольку не все компьютеры раньше имели клавиатуры с клавишами фигурных скобок. Но и поныне команды `\lbrace`, `\rbrace`, `\vert` и `\Vert` могут быть полезны, например, при составлении алфавитного указателя, так как на использование символов `{`, `}` и `|` в аргументе команды `\index` существует ряд ограничений (глава 14).

6.4. Основные структуры

6.4.1. Индексы

Верхний индекс вводит команда `^`, а нижний — команда `_`. Они имеют по одному аргументу:

`^{`*superscript*`}`	`\sp{`*superscript*`}`
`_{`*subscript*`}`	`\sb{`*subscript*`}`

В редких случаях (например, в процедуре `alltt`) вместо `^` и `_` приходится использовать команды `\sp` и `\sb`, которые чуть менее удобны. Фигурные скобки обязательны, если в индекс необходимо вставить несколько символов:

`$x^{2y}$`	x^{2y}	`$x_{y_2}$`	x_{y_2}
`$x^{2^y}$`	x^{2^y}	`$x^{2n}_{i}$`	x^{2n}_{i}

Если индекс состоит из одного символа или одной команды, то фигурные скобки можно опустить:

`$x^2$`	x^2	`$2^\alpha$`	2^α
`$x_2$`	x_2	`$x ^ 2y ^ 2$`	x^2y^2

Здесь `^` и `_` действуют только на следующий символ, отличный от пробела. Неверно набирать `x^y^z` или `x_y_z`, так как LaTeX не признаёт двойных индексов. Однако он поймёт выражение типа `x^{y^z}` или `{x^y}^z`.

```
Следует различать $x^{y^z}$ и
${x^y}^z$.
```

Следует различать x^{y^z} и ${x^y}^z$.

По умолчанию LaTeX использует для индексов шрифт меньшего размера, чем для основной формулы, а индекс у индекса печатает ещё более мелким шрифтом. Как изменить размер шрифта в формуле, мы расскажем в разделе 6.5.

Индекс, следующий за символом, относится только к этому символу, но если он следует за группой символов, то он относится ко всей группе. Группа (блок)

внутри формулы выделяется по общему правилу (раздел 2.4) при помощи фигурных скобок:

```
$((x^2)^3)^4$ и ${({(x^2)}^3)}^4$
```
$((x^2)^3)^4$ и ${({(x^2)}^3)}^4$

Изредка случается, что индекс ни за чем не следует, как в обозначении гипергеометрической функции ${}_1F_2$. В этом случае лучше всего предпослать индексу пустой блок или всё обозначение окружить фигурными скобками, чтобы сделать свои намерения ясными также и LaTeX'у. Иными словами, лучше всего набирать обозначение ${}_1F_2$ в виде `${}_1F_2$`, или `${_1}F_2$`, или `${_1F_2}$`. Следующий пример показывает, что такая предосторожность не лишена смысла:

```
Сравните: $G+_1F_2$ и $G+{}_1F_2$
```
Сравните: $G+_1F_2$ и $G+{}_1F_2$

Верхние и нижние индексы могут использоваться вместе, причём порядок их следования в исходном тексте безразличен:

```
$x^2_3 = x_3^2$
```
$x^2_3 = x_3^2$

Верхние и нижние индексы обычно позиционируются друг над другом. Однако верхний индекс немного смещается, если следует за некоторыми буквами. Так, `$P_2^2$` производит P_2^2. Если по каким-то причинам необходимо всё-таки выровнять левые края индексов, то это достигается при помощи пустого блока: `$P{}_2^2$` производит $P{}_2^2$. Тот же приём помогает, когда требуется жёстко выдерживать порядок следования верхних и нижних индексов. Такое случается в публикациях, использующих формулы тензорного анализа:

```
$A^{i}{}_{jk}{}^{l}$
```
$A^{i}{}_{jk}{}^{l}$

Математики часто используют символ «штрих» ($\prime$) в верхнем индексе для обозначения производной. Например, вторая производная функции f обозначается как f''. Изучив таблицу 6.4, Читатель сможет найти, что подходящий символ для штриха производит команда `\prime`:

```
$f^{\prime\prime\prime}$
```
$f^{\prime\prime\prime}$

Предусмотрен и более экономный способ расстановки штрихов при помощи апострофа (`'`). С его помощью f''' набирается в виде `$f'''$`. Использование апострофа не мешает применению других индексов, как показывает следующий пример:

```
$ y'_1 + y''_2 = g'^2 $
```
$y'_1 + y''_2 = g'^2$

Однако лучше сразу привыкнуть к более строгой форме записи, явно указывая, к чему относится второй верхний индекс:

```
Сравните: $y'_1+y''_2  =g'{}^2$
 и        $y'_1+{y'}'_2={g'}^2$.
```
Сравните: $y'_1+y''_2=g'{}^2$ и $y'_1+{y'}'_2={g'}^2$.

Мы уже упоминали, что индексы у некоторых символов переменного размера могут располагаться не на обычном месте справа от изображения символа, а над или под ним, если символ используется в выключной формуле[4]:

[4] Точнее, при использовании декларации `\displaystyle` (раздел 6.5).

```
Пример символов переменного
размера в выключной формуле
\[
   \int_{a}^{b}y\,dx =
   h\sum_{i=0}^{n-1}y_{i}
\]  и в строке
   $\int_{a}^{b}y\,dx=
   h\sum_{i=0}^{n-1}y_{i}$.
```

Пример символов переменного размера в выключной формуле

$$\int_{a}^{b}y\,dx = h\sum_{i=0}^{n-1}y_{i}$$

и в строке $\int_{a}^{b}y\,dx= h\sum_{i=0}^{n-1}y_{i}$.

То же самое происходит с индексами у обозначений некоторых функций. Полное перечисление таких символов и функций лишено смысла. Во-первых, потому, что выбор положения индексов, предлагаемый LaTeX'ом по умолчанию, покажется разумным большинству математиков. Во-вторых, этот выбор можно изменить при помощи команд

```
\limits
\nolimits
```

Их нужно поставить после символа перед индексами. Команда `\limits` указывает, что индексы нужно позиционировать над и под символом; команда `\nolimits` имеет обратное действие. В следующем примере эти команды изменяют режим позиционирования индексов у символов $\int$ и $\sum$ на противоположный принятому по умолчанию:

```
\[ \int\limits_0^\infty\sum\nolimits_{n=1}^m \]
```

$$\int\limits_0^\infty\sum\nolimits_{n=1}^m$$

В формулах, размещаемых внутри абзаца, использование `\nolimits` редко имеет смысл[5], так как и без неё индексы располагаются сбоку от любых символов. Команду `\limits` можно использовать, если смириться с тем, что она обычно приводит к увеличению интервала между соседними строками. Так, формулу $\lim\limits_{x\to 0}\sin(x)=0$ можно получить, набрав `$\lim\limits_{x\to 0}\sin(x)=0$`. При этом эстетическому восприятию текста будет нанесен определённый ущерб.

6.4.2. Дроби

Дроби создаются символом `/` или командой

`\frac{`*числитель*`}{`*знаменатель*`}`

Наклонная дробная черта используется, главным образом, в формулах, размещаемых внутри абзаца:

```
Деление на $n/2$ даёт \( (m+n)/n \).
```

Деление на $n/2$ даёт $(m+n)/n$.

[5] Разве что если явно указана декларация `\displaystyle` (раздел 6.5).

Команда `\frac` обычно применяется в выключных формулах, размещаемых отдельной строкой:

```
\[ x=\frac{y^2+z/3}{4+\frac{y}{z+a}} \]
```

$$x = \frac{y^2 + z/3}{4 + \frac{y}{z+a}}$$

Её можно использовать также для формул в абзаце, чтобы получить дробь типа $\frac{1}{2}$, но лучше писать $1/2$, а также избегать многоэтажных дробей, если на то нет особых оснований. Когда числитель и/или знаменатель состоят из одного символа, фигурные скобки у соответствующего аргумента можно опустить. Например, $\frac{1}{2}$ можно набрать посредством `$\frac12$` или `$\frac 1 2$`, хотя `\frac{1}{2}` лучше отражает структуру формулы.

Изображение дроби зависит от того, находится ли она в обычной или выключной формуле, под знаком корня или «внутри» другой дроби. Как изменить правила форматирования, используемые по умолчанию, показано в разделе 6.5. В разделе 8.11.11 описаны ещё несколько команд, которые вводит пакет amsmath для набора дробей. С их помощью, например, можно задать толщину дробной черты, если это зачем-то нужно.

6.4.3. Корни

Команда

⚠ `\sqrt[`*n*`]{`*math*`}`

генерирует обозначение корня из выражения, стоящего в аргументе *math*. При наличии опции *n* команда `\sqrt` производит корень степени n:

```
$\sqrt{2}$
$\sqrt{2+x}$
$\sqrt[3]{x^2+\sqrt{\alpha}}$
```

$\sqrt{2}$
$\sqrt{2+x}$
$\sqrt[3]{x^2+\sqrt{\alpha}}$

В простых случаях, когда под знак корня необходимо поместить только один символ, фигурные скобки, окружающие обязательный аргумент команды `\sqrt`, можно опустить. Например, $\sqrt{2}$ можно набрать в виде `\sqrt2`, а $\sqrt{\alpha}$ — в виде `\sqrt\alpha`. Правильной является также запись `\sqrt x` для $\sqrt{x}$, но `\sqrt{x}` яснее отражает структуру формулы.

LaTeX позиционирует знак корня в соответствии с размерами подкоренного выражения: $\sqrt{a} + \sqrt{d} + \sqrt{g}$. Если в формуле имеется только один знак корня, правила позиционирования, используемые LaTeX'ом, работают идеально, но при наличии нескольких корней, возможно, следует предпочесть более однородный формат $\sqrt{\mathstrut a} + \sqrt{\mathstrut d} + \sqrt{\mathstrut g}$. Соответствующая подгонка размеров знака корня выполняется методами, описанными в разделе 6.9.

6.4.4. Размещение объектов друг над другом

Символы в математических формулах иногда помещают друг над другом. Простейший пример дают символы с диакритическими знаками. При отсутствии

подходящих диакритических знаков нужные комбинации можно получить при помощи команды

```
\stackrel{top}{bottom}
```

Она печатает текст из первого аргумента top непосредственно над текстом из второго аргумента bottom, причём top печатается шрифтом меньшего размера (как индексы):

```
$\stackrel{\Leftrightarrow}{A}$
```

$\stackrel{\Leftrightarrow}{A}$

Следующий пример показывает, что команда \stackrel может применяться для построения более сложных конструкций:

```
\( A \stackrel{a'}{\rightarrow} D \)
```

$A \stackrel{a'}{\rightarrow} D$

Наконец, с её помощью удобно давать определения математическим обозначениям:

```
\( \vec{v} \stackrel{\mathrm{def}}{\equiv}
   (v_x,v_y,v_z) \)
```

$\vec{v} \stackrel{\mathrm{def}}{\equiv} (v_x,v_y,v_z)$

Здесь полезно обратить внимание на первое в нашей практике явное переключение шрифта в формуле. Чтобы набрать «def» прямым шрифтом, вместо известной Читателю команды \textrm использована команда \mathrm, поскольку первая не может применяться в математической моде (см. раздел 6.6).

Команды

```
\overline{math}
\underline{math}
```

используются соответственно для *надчеркивания* и *подчеркивания* формулы, стоящей в math, на всю её длину:

```
$\overline{x^2+\overline{y}}=\underline{5z}$
```

$\overline{x^2+\overline{y}}=\underline{5z}$

Команда \underline работает также вне математической моды. Команды

```
\overbrace{math}
\underbrace{math}
```

соответственно вставляют над и под аргументом math горизонтальные фигурные скобки:

```
$ \underbrace{ a + \overbrace{b+c} + d } $
```

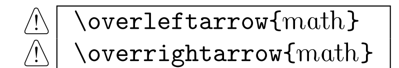
$\underbrace{ a + \overbrace{b+c} + d }$

Верхние и нижние индексы к этим командам размещаются в виде меток фигурных скобок:

```
$\underbrace{a+
 \overbrace{b+\cdots+c}^{5}+d}_{7}$
```

$\underbrace{a+\overbrace{b+\cdots+c}^{5}+d}_{7}$

Команды



```
\overleftarrow{math}
\overrightarrow{math}
```

рисуют длинные стрелки над объектом:

```
$\overleftarrow{ABC+\overrightarrow{abc}}$
```

$\overleftarrow{ABC+\overrightarrow{abc}}$

6.4.5. Матрицы

Матрицы создаются процедурой

```
\begin{array}[hpos]{cols}   ...   \end{array}
```

Её обязательный аргумент *cols* указывает количество столбцов (колонок) в матрице. Каждой колонке в аргументе процедуры `array` отвечает одна буква, которая указывает, как позиционируются формулы в каждой колонке:

`l` — выравнивание по левой границе колонки,
`r` — выравнивание по правой границе колонки,
`c` — выравнивание по центру колонки.

Необязательный аргумент *hpos* указывает способ вертикального позиционирования матрицы:

`t` — выравнивание по верхней строке,
`c` — выравнивание по центру матрицы (используется по умолчанию),
`b` — выравнивание по нижней строке.

Последовательные строки в теле процедуры разделяются командами `\\`, а элементы колонок в строке — символами `&`:

```
\[ \begin{array}{lcr}
a+x-y & b   & 4x  \\
x+y   & 2+5 & a+b \\
x     & xz  &  -4
\end{array} \]
```

$$\begin{array}{lcr}
a+x-y & b & 4x \\
x+y & 2+5 & a+b \\
x & xz & -4
\end{array}$$

Вслед за последним элементом колонки в строке не должно быть `&`, а за последней строкой не должно быть `\\`, иначе LaTeX будет думать, что строк и колонок больше, чем нужно. LaTeX находится в математической моде, когда обрабатывает элементы колонок, поэтому все пробелы он игнорирует. В матрицах элементы колонок обычно центрируются (спецификатор `c` в аргументе процедуры), однако колонка чисел выглядит лучше, если числа сдвинуты вправо (спецификатор `r`). Декларации, которые могут находиться в элементах колонок, являются локальными, то есть область их действия оканчивается символом `&`, командой `\\` или командной скобкой `\end{array}`.

Через каждую формулу LaTeX проводит воображаемую горизонтальную осевую линию там, где должен стоять знак минус. Объединяя несколько формул в

одну, LaTeX располагает их осевые линии на одном уровне. По умолчанию в матрице осевая линия проходит через её центр, а элементы матрицы в каждой строке выравниваются так, чтобы их осевые линии находились на одном уровне. При наличии опции `t` осевая линия матрицы совмещается с осевой линией верхней строки, а при наличии опции `b` — с осевой линией нижней строки. В приводимом ниже примере[6] матрицы, создаваемые каждой из трёх процедур `array`, для лучшего понимания обведены рамкой:

```
\[ x-
\begin{array}{c}
 a_1 \\ \vdots\\ a_n
\end{array} -
\begin{array}[t]{cl}
 x-y & 21 \\
 a+b & \begin{array}[b]{r}
        16 \\ -204
       \end{array}
\end{array}  \]
```

$$x - \boxed{\begin{array}{c} a_1 \\ \vdots \\ a_n \end{array}} - \boxed{\begin{array}[t]{cl} x-y & 21 \\ a+b & \boxed{\begin{array}[b]{r} 16 \\ -204 \end{array}} \end{array}}$$

Матрицы часто окружают большими скобками или другими разделителями. Команды `\left` и `\right` позволяют автоматически подобрать размер разделителей, окружающих матрицу:

```
\[ \left( \begin{array}{c}
   \left|  \begin{array}{cc}
             a & b \\ c & d
           \end{array}
   \right|\\ x \\ y
    \end{array} \right) \]
```

$$\left(\begin{array}{c} \left| \begin{array}{cc} a & b \\ c & d \end{array} \right| \\ x \\ y \end{array} \right)$$

Часто процедура `array` вместе с большими разделителями решает все проблемы там, где, казалось бы, нужны другие средства. В следующем примере команда `\right.` (с точкой на конце!) создаёт невидимый разделитель:

```
\[ \sigma(x) = \left\{
\begin{array}{rl}
-1, & \mbox{если } x>0 \\
 0, & \mbox{если } x=0 \\
 1, & \mbox{если } x<0
\end{array} \right. \]
```

$$\sigma(x) = \left\{ \begin{array}{rl} -1, & \mbox{если } x>0 \\ 0, & \mbox{если } x=0 \\ 1, & \mbox{если } x<0 \end{array} \right.$$

Чтобы придать матрице идеальный вид, иногда необходимо прибегать к визуальному форматированию. Например, изменить расстояние между строками можно при помощи необязательного аргумента у команды `\\` (раздел 4.6). Изменить горизонтальные промежутки между колонками можно командами из раздела 6.7. Имеется и более регулярный метод, когда расстояние между колонками указывается в аргументе cols процедуры `array` (глава 12).

[6] Абсолютно бессмысленном с научной точки зрения.

6.5. Стиль формулы

Внимательный Читатель имел возможность заметить, что буквы и другие символы уменьшаются в размерах, когда появляются в дробях, под знаком корня или индексах. Теперь пришло время объяснить, как LaTeX выбирает размер объектов в математических формулах.

Существуют восемь стилей форматирования математических выражений. Четыре из них являются основными. Для краткости их принято обозначать D, T, S и SS. Они вводятся, соответственно, декларациями

```
\displaystyle
\textstyle
\scriptstyle
\scriptscriptstyle
```

По умолчанию стиль D применяется в выключных формулах, стиль T — для формул внутри абзаца, S — для индексов, SS — для индексов в индексах. Ещё четыре редуцированных варианта этих стилей D', T', S', SS' вводятся теми же декларациями, но используются для форматирования математических выражений, которые должны быть ограничены по высоте. Например, редуцированный стиль D' используется для форматирования выражения, стоящего под знаком корня или в знаменателе дроби в выключной формуле. Аналогично стиль T' используется в тех же случаях, но в формуле внутри абзаца.

Для всех восьми стилей LaTeX использует три размера букв, цифр и других математических символов T, S и SS:

Символ в стиле	печатается размером	
D, D', T, T'	T	(примерно так)
S, S'	S	(примерно так)
SS, SS'	SS	(примерно так)

Если формула напечатана, скажем, в стиле D или T, то верхний индекс печатается в стиле S, а нижний — в стиле S'. Редуцированные варианты математических стилей отличаются более компактным расположением индексов. При необходимости правила форматирования формул можно изменить при помощи указанных выше деклараций. Например:

```
индексный стиль \( e^{f(x)} \)
можно заменить текстовым
\( e^{\textstyle f(x)}.
```

индексный стиль $e^{f(x)}$ можно заменить текстовым $e^{\textstyle f(x)}$.

Впечатляющий пример использования деклараций математического стиля дают так называемые «непрерывные дроби». Простое вложение команды `\frac` саму в себя производит результат, который вряд ли удовлетворит эстета:

```
\[x+\frac{1}{x+
    \frac{1}{x+
    \frac{1}{x+
    \frac{1}{x}}}} \]
```

$$x+\frac{1}{x+\frac{1}{x+\frac{1}{x+\frac{1}{x}}}}$$

Небольшое усовершенствование приводит к желаемому результату:

```
\[x +       \frac{1}{x+
\displaystyle\frac{\mathstrut 1}{x+
\displaystyle\frac{\mathstrut 1}{x+
\displaystyle\frac{\mathstrut 1}{x}}}}\]
```

$$x+\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x+\cfrac{1}{x}}}}$$

Здесь команда `\mathstrut` (раздел 6.9) поставлена в числители дробей, чтобы сделать их повыше.

6.6. Шрифты

По умолчанию в математических формулах буквы печатаются в курсивном, а цифры — в прямом начертании. LaTeX расставляет символы в формулах менее плотно по сравнению с обычным текстом, а лигатуры вообще отсутствуют. Декларации переключения гарнитуры шрифта типа `\itshape` или `\rmfamily` и соответствующие им команды `\textit` и `\textrm` нельзя использовать в формулах[7]. Более того, эти декларации и команды не действуют даже тогда, когда весь текст напечатан, скажем, полужирным шрифтом:

```
Сравните: 2f(x), $2f(x)$ и
  \textbf{2f(x), $2f(x)$}.
```

Сравните: 2f(x), $2f(x)$ и **2f(x),** $2f(x)$.

В редких случаях, когда действительно необходимо изменить шрифт в формуле, следует использовать команды, которые называются *математическими алфавитами*:

`\mathrm{letters,etc}`	$\mathrm{ABCabc}, \mathrm{123}, \mathrm{\hat{a}}, \mathrm{\tilde{b}}, \mathrm{\tilde{c}}, \mathrm{\Psi\Omega}$
`\mathbf{letters,etc}`	$\mathbf{ABCabc}, \mathbf{123}, \mathbf{\hat{a}}, \mathbf{\tilde{b}}, \mathbf{\tilde{c}}, \mathbf{\Psi\Omega}$
`\mathsf{letters,etc}`	$\mathsf{ABCabc}, \mathsf{123}, \mathsf{\hat{a}}, \mathsf{\tilde{b}}, \mathsf{\tilde{c}}, \mathsf{\Psi\Omega}$
`\mathtt{letters,etc}`	$\mathtt{ABCabc}, \mathtt{123}, \mathtt{\hat{a}}, \mathtt{\tilde{b}}, \mathtt{\tilde{c}}, \mathtt{\Psi\Omega}$
`\mathit{letters,etc}`	$\mathit{ABCabc}, \mathit{123}, \mathit{\hat{a}}, \mathit{\tilde{b}}, \mathit{\tilde{c}}, \mathit{\Psi\Omega}$
`\mathnormal{letters,etc}`	$\mathnormal{ABCabc}, \mathnormal{123}, \mathnormal{\Psi\Omega}$
`\mathcal{capital letters}`	$\mathcal{ABC}$

Они действуют на буквы, цифры, диакритические знаки и прописные греческие буквы:

```
$\mathbf{\tilde{A} \times 2 = 2\Phi}$
```

$\mathbf{\tilde{A} \times 2 = 2\Phi}$

Другие символы, как $\times$ и $=$, не меняются.

Особую статью составляют алфавиты `\mathcal` и `\mathnormal`. Первый из них содержит только двадцать шесть прописных букв английского алфавита $\mathcal{A}, \mathcal{B}, \ldots, \mathcal{Z}$. Второй печатает цифры в так называемом «старом стиле» и почему-то путает некоторые диакритические знаки.

[7] Точнее сказать, результат действия этих команд в математической моде не гарантирован.

Русские буквы в формулах исчезают, если не принять особых мер. Когда в 1998 году появилась стандартная русификация LaTeX'а, для многих пользователей это стало чуть ли не трагедией, но сейчас жалоб уже не слышно. Русские буквы в формулах можно получить с помощью команды `\mbox`. Она печатает свой аргумент в текстовом режиме, поэтому там действуют команды переключения шрифтов типа `\text`*cmd*, а по умолчанию используется тот шрифт, которым набран текст непосредственно перед формулой[8]:

```
$\mbox{Я} = 10^{81}$
```

$\mbox{Я} = 10^{81}$

Но такой вариант имеет ряд изъянов. В частности, размер букв в индексах не будет автоматически уменьшен, как для символов, не находящихся внутри `\mbox`:

```
$M_{\mbox{крит}} \neq M_{crit}$
```

$M_{\mbox{крит}} \neq M_{crit}$

Однако вместо `\mbox` можно воспользоваться командой `\text`:

```
$M_{\text{крит}} \neq M_{crit}$
```

$M_{\text{крит}} \neq M_{crit}$

Она определена в пакете amsmath, которому мы посвятим целиком главу 8. Есть также малоизвестный пакет mathtext Владимира Воловича. Он распространяется в составе коллекции T2 и должен быть загружен до любого из пакетов, которые влияют на шрифты, в том числе fontenc, inputenc, babel. Загрузив пакет mathtext, можно использовать русские буквы в математических формулах почти наравне с латинскими. Почти — потому что русские буквы появляются только вне математических алфавитов (в аргументах команд `\math`*cmd* они по-прежнему игнорируются) и к тому же отображаются прямым шрифтом, а не курсивом, как латинские. Как решить проблему русских букв в математических алфавитах, мы расскажем в разделе 16.6.

Математические шрифты, как и текстовые, различаются по гарнитуре, насыщенности, начертанию и размеру. Однако не существует команд, которые бы переключали только один атрибут у шрифтов в формулах. Поэтому комбинация математических алфавитов в отличие от их текстовых аналогов не даёт новый шрифт:

```
$\mathbf{\mathit{A}}=\mathit{A},
 \mathit{\mathbf{A}}=\mathbf{A}$.
```

$A = A, \mathbf{A} = \mathbf{A}$.

Вместо такого комбинирования соответствие каждого алфавита шрифту с заданным перечнем атрибутов контролирует *математическая версия*. Версию изменяет декларация `\mathversion{`*version-name*`}`, где *version-name* — название версии. Её можно использовать только вне математической формулы. Стандартные классы печатного документа предопределяют две версии: `normal` (используется по умолчанию) и `bold`. Изменение версии влияет не только на буквы, цифры и диакритические знаки, но и на другие символы.

```
Сравните:  $\mathcal{X}^2+\sqrt{2\pi y}$ и
{\mathversion{bold} $\mathcal{X}^2+\sqrt{2\pi y}$}.
```

Сравните: $\mathcal{X}^2 + \sqrt{2\pi y}$ и $\boldsymbol{\mathcal{X}^2 + \sqrt{2\pi y}}$.

[8] 10^{81} — это приблизительное число частиц во Вселенной.

Поскольку область действия `\mathversion` может быть указана фигурными скобками, как в примере выше, декларация `\mathversion{normal}` используется редко. При необходимости набрать полужирным математическим курсивом только некоторые из символов, можно поступить так, как предлагает следующий пример:

```
$\mbox{div\mathversion{bold}$B$} = 0$
```

div$\boldsymbol{B} = 0$

Аргумент команды `\mbox` печатается шрифтом с гарнитурой и кеглем, которые использовались непосредственно перед переходом в математическую моду.

Более качественное решение даёт команда

`\bm{`*math*`}` (bm)

которую вводит одноимённый пакет bm из коллекции tools. Она печатает выражение *math* полужирным шрифтом, сохраняя его правильное позиционирование относительно других частей формулы:

```
Сравните $A \otimes B$, $A \bm{\otimes} B$
и $A \mbox{\mathversion{bold}$\otimes$} B$.
```

Сравните $A \otimes B$, $A \boldsymbol{\otimes} B$ и $A\boldsymbol{\otimes}B$.

6.7. Пробелы в формулах

LaTeX знает о правилах оформления математических формул гораздо больше, чем большинство авторов этих формул. Поэтому, прежде чем улучшать внешний вид формулы, всегда полезно сначала посмотреть, как её сформатирует LaTeX. Тем не менее можно дать несколько общих рекомендаций, когда такое улучшение необходимо.

Первая рекомендация относится к знакам пунктуации в математических формулах. Таких знаков всего два: «,» и «;» — плюс ещё три команды, перечисленные в табл. 6.10. LaTeX вставляет небольшой пробел после знака пунктуации, но не перед ним. Этот пробел меньше, чем в обычном тексте. Поэтому, если формула предназначена для размещения внутри абзаца, знаки пунктуации лучше выносить за пределы формул:

```
Функция $F_1(x,y;z)$ зависит
от $x$, $y$, $z$.
```

Функция $F_1(x,y;z)$ зависит от x, y, z.

LaTeX рассматривает точку как обычный символ и поэтому не увеличивает промежуток между ней и следующим символом. Если запятая используется в качестве обычного символа (например, в десятичном числе), её необходимо окружить фигурными скобками:

```
$12,345$  (неправильно)\\
$12{,}345$ (правильно)
```

$12, 345$ (неправильно)
$12{,}345$ (правильно)

Есть более регулярный способ разрешения подобных проблем: в дополнение к рассмотренным в разделе 4.3 существуют ещё шесть команд для управления горизонтальными пробелами:

`\,`	⌋⌈	узкий	`\:`	⌋ ⌈	средний
`\;`	⌋ ⌈	широкий	`\!`	⌋⌈	отрицательный
`\quad`	⌋ ⌈	очень широкий	`\qquad`	⌋ ⌈	ещё шире

Все эти команды могут использоваться как в формулах, так и в обычном тексте.

Корректировка промежутков необходима, если LaTeX не способен распознать логическую структуру формулы. Например, выражение `$y dx$` интерпретируется им как произведение трёх величин: y, d и x, а не двух: y и dx, поэтому LaTeX не увеличит промежуток между y и dx, если не сделать это принудительно посредством команды `\,`. Особое внимание следует обратить на знаки корня, знаки интеграла и суммы. Сравните:

`\sqrt{\pi} \, y`	$\sqrt{\pi}\, y$	и	$\sqrt{\pi} y$
`n / \! \log n`	$n/\!\log n$	и	$n/\log n$
`\int\!\!\int f(x,y)\,dx\,dy`	$\int\!\!\int f(x,y)\,dx\,dy$	и	$\int\int f(x,y)dxdy$

Здесь в последней колонке приведен пример формул, в которых корректировка промежутков не производилась.

6.8. Многострочные формулы

Процедуры

`\begin{eqnarray}`	*eqns*	`\end{eqnarray}`
`\begin{eqnarray*}`	*eqns*	`\end{eqnarray*}`

предназначены для печати систем уравнений или длинных формул, которые не умещаются в одной строке. Они производят последовательность выключных формул, выровненных в трёх колонках. Тело процедур должно быть подготовлено, как для процедуры `array`[9] с аргументом `rcl`. Оно состоит из последовательности строк, разделённых командами `\\`, а каждая строка состоит из трёх колонок, разделённых при помощи служебного символа `&`. В первой и третьей колонках математические выражения форматируются в стиле D, как в однострочных выключных формулах, а в средней колонке — в стиле T, как в формулах внутри абзаца (в ней обычно ставят знак равенства или другой оператор сравнения). В отличие от процедуры `array`, которая используется в математической моде, процедуры `eqnarray` и `eqnarray*` работают только в текстовой моде, так как они сами переводят LaTeX в математическую моду. В процедуре `eqnarray` каждая строка нумеруется, если в ней отсутствует команда

```
\nonumber
```

которая подавляет нумерацию:

[9] Однако команду `\multicolumn` (глава 12) использовать нельзя.

```
\begin{eqnarray}
 x& = & 21y\\
 y& < & a+b+c+ \nonumber \\
  &   & d - e
\end{eqnarray}
```

$$\begin{eqnarray} x & = & 21y \\ y & < & a+b+c+ \nonumber \\ & & d-e \end{eqnarray}$$ (6.2), (6.3)

Ссылку на любую строку уравнения, как обычно, можно сделать при помощи команды `\label`. Её целесообразно размещать либо в начале, либо в конце соответствующей строки. В `*`-форме процедуры `eqnarray` нумерация отсутствует.

Если при расщеплении формулы на несколько строк новая строка должна начинаться со знака бинарной операции (`+`, `-` и т. д.), то перед ним полезно вставить команду `\mbox{}`, иначе LaTeX будет рассматривать знак бинарной операции как обычный символ и удалит дополнительный пробел после него. При форматировании очень длинных формул полезной оказывается также команда

`\lefteqn{`*math*`}`

Её используют обычно в левой колонке, помещая весь текст в её аргумент *math*. Он будет сформатирован обычным образом, но LaTeX будет считать, что длина формулы в колонке равна нулю. Поэтому колонка будет сделана предельно узкой. Если в многострочной формуле используются большие скобки, увеличенные при помощи `\left` и `\right`, то в каждой строке эти команды должны использоваться попарно. Следующий пример иллюстрирует все сказанное.

```
\begin{eqnarray*}
\lefteqn{x+y+z=}              \\
& & \left(a+b+2c     \right. \\
& & \left.\mbox{}-d+m\right)
\end{eqnarray*}
```

$$\begin{eqnarray*} \lefteqn{x+y+z=} \\ & & \left(a+b+2c \right. \\ & & \left.\mbox{}-d+m\right) \end{eqnarray*}$$

Здесь команда `\mbox` использована, чтобы «оттянуть» знак минус от первого слагаемого в начале последней строки. Поскольку всё познается в сравнении, покажем, что получается с тем же уравнением без применения «ухищрений» в виде команд `\lefteqn` и `\mbox`:

```
\begin{eqnarray*}
x+y+z &=                      \\
& & \left(a+b+2c     \right. \\
& & \left.-d+m\right)
\end{eqnarray*}
```

$$\begin{eqnarray*} x+y+z &= \\ & & \left(a+b+2c \right. \\ & & \left.-d+m\right) \end{eqnarray*}$$

Вместо `eqnarray` можно порекомендовать процедуры из раздела 8.6, которые вводит пакет `amsmath`. Если нашему Читателю часто приходится иметь дело с громоздкими формулами в своих публикациях, он должен получить второе высшее математическое образование, прочитав главу 8.

6.9. Позиционирование в формулах

Выполним обещание, данное в разделе 6.4.3, и расскажем, как сделать одинаковыми знаки корня в формуле $\sqrt{a}+\sqrt{d}+\sqrt{g}$. Ответ в принципе прост: нужно заставить LaTeX «думать», будто высота подкоренного выражения везде одинакова. Для этого достаточно вставить в подкоренное выражение невидимый символ достаточно большой *высоты* и *глубины*. Высота — это расстояние, на которое символ или целое выражение возвышается над невидимой осевой линией. Соответственно, глубина — это расстояние от осевой линии до нижнего края выражения. В математической формуле осевая линия проходит горизонтально там, где находится или мог бы находиться знак минус. В данном случае самой высокой буквой является d, а ниже всего от осевой линии спускается «хвостик» буквы g. Обычная круглая скобка чуть выше, чем d, и опускается чуть «глубже», чем g, а команда

```
\mathstrut
```

как раз вставляет невидимый символ, имеющий высоту круглой скобки и нулевую ширину. Такие невидимые символы называются стратами. Так что искомое решение может быть таким:

```
$ \sqrt{a\mathstrut} +
  \sqrt{d\mathstrut} +
  \sqrt{g\mathstrut} $
```

$\sqrt{a\mathstrut}+\sqrt{d\mathstrut}+\sqrt{g\mathstrut}$

В общем случае вместо `\mathstrut` можно использовать команду

```
\vphantom{math}
```

Она создаёт страту, которая простирается вверх и вниз от осевой линии на такие же расстояния, как и формула {*math*}; сама формула при этом не печатается. Обратите внимание, что в следующем примере знак корня увеличивается только вверх, но не вниз.

```
$ \sqrt{a\vphantom{d}} + \sqrt{d}
```

$\sqrt{a\vphantom{d}}+\sqrt{d}$

В многострочных формулах взаимное расположение выражений в разных строках может быть отрегулировано при помощи команды

```
\phantom{math}
```

Она создаёт пробел с шириной, равной ширине невидимой формулы, набранной в виде {*math*} (вместе с фигурными скобками).

```
\begin{eqnarray*}
W &=& \sum a_i+E+F\\
  & & \phantom{\sum a_i}+G
\end{eqnarray*}
```

$$\begin{aligned} W &= \sum a_i+E+F\\ & +G \end{aligned}$$

Задачу, обратную `\phantom` и `\vphantom`, решает команда

`\smash{math}`

Она форматирует формулу {math}, но заставляет LaTeX думать, будто та имеет нулевую высоту и глубину. Мы ещё поговорим о её усовершенствованном варианте, который имеется в пакете amsmath (раздел 8.11.12).

6.10. Параметры настройки

`\jot` — высота пробела между строками в процедурах `eqnarray` и `eqnarray*`. Нерастяжимая длина.

`\mathindent` — расстояние между левой границей окружающего текста и левым краем формулы при использовании в `\documentclass` опции `fleqn`, которая устанавливает выравнивание выключных формул по левому краю страницы, а не по центру строк. Нерастяжимая длина.

`\abovedisplayskip` — дополнительный вертикальный пробел над длинными выключными формулами, которые начинаются левее конца предыдущей строки. Если выключные формулы выравниваются по левому краю страницы, взамен `\abovedisplayskip` используется `\topsep`. Растяжимая длина.

`\belowdisplayskip` — дополнительный пробел под длинными выключными формулами. Не работает с опцией `fleqn`, использующей команду `\topsep`. Растяжимая длина.

`\abovedisplayshortskip` — дополнительный пробел над короткими выключными формулами, которые начинаются правее того места, где окончилась предыдущая строка. Не работает с опцией `fleqn`, использующей команду `\topsep`. Растяжимая длина.

`\belowdisplayshortskip` — дополнительный пробел под короткими выключными формулами. Не работает с опцией `fleqn`, использующей команду `\topsep`. Растяжимая длина.

Сведений, изложенных в этой главе, более чем достаточно для подготовки математически очень сложных текстов. Если же наступит день, когда что-то не удастся сделать изложенными здесь методами,— обратитесь к главе 8, посвящённой коллекции пакетов AMS-LaTeX. Пока же мы прервём урок математики и в следующей главе займемся конструированием новых команд и процедур.

Если всё иное бесполезно,
читайте инструкцию.
Аксиома Кана

Глава 7

Программируйте сами

Подготовка печатного документа из рутинного дела может стать увлекательным занятием, если включить на полную мощь такой универсальный инструмент, как программирование команд и процедур. Мы коснёмся этой темы лишь настолько, насколько программирование способно облегчить участь рядового пользователя LaTeX'а. Например, в предыдущей главе мы писали, что длинные названия математических символов можно сократить, введя команды-синонимы, хотя мы сами этого никогда не делаем, предпочитая ясность краткости. Для тех, кто владеет английским, длинные имена математических команд даже удобны, так как обычно совпадают с названиями математических символов. Однако более сложную структуру, которая часто повторяется, действительно бывает полезно представить в виде команды с кратким удобным именем, проясняющим её назначение. LaTeX позволяет переопределять существующие и создавать новые команды и процедуры. Данная глава рассказывает, как это следует делать. В разделе 7.3 вводится новое понятие — теорема. Это особая процедура, которая автоматически печатает заголовок, общий для всех теорем одного вида, и порядковый номер. Так в учебниках по математике обычно оформляются теоремы, леммы и другие подобные утверждения.

7.1. Определение новых команд

Если в документе часто используется одно и то же словосочетание, например «Г-н Председатель», то с помощью `\newcommand` можно определить коротенькую команду, которая будет генерировать этот незамысловатый текст:

```
\newcommand{\boss}{Г-н Председатель}
\boss\ заявил, что не может быть двух
мнений относительно плюрализма мнений.
\boss\ также сказал то, что думает.
```

Г-н Председатель заявил, что не может быть двух мнений относительно плюрализма мнений. Г-н Председатель также сказал то, что думает.

Очень часто декларация `\newcommand` используется для сокращения записи математических формул:

```
\newcommand{\gn}{$\alpha_{v}$}.
Пусть \gn\ есть число мух в единице
объёма, причём при подсчёте \gn\
учитываются особи только мужского пола.
```

Пусть α_v есть число мух в единице объёма, причём при подсчёте α_v учитываются особи только мужского пола.

Нововведённая команда `\gn` используется здесь для сокращения записи математических формул, заменяя собой символ α с индексом v. Напомним, что пробел после имени команды игнорируется. Поэтому после `\gn` вставлена команда `\␣`, принудительно создающая пробел. В конце этого раздела мы покажем, как создать команду, которая не игнорирует пробелы после себя, хотя существование подобной команды противоречило бы практике программирования для LaTeX'а.

Приведённый пример иллюстрирует общую проблему с созданием новых команд. В данном случае команда `\gn` предназначена для использования в текстовом или строковом режимах. Она содержит команду `\alpha`, которая может применяться только в математическом режиме, и именно поэтому значки `$` присутствуют в определении команды `\gn` в аргументе `\newcommand`, окружая `\alpha` с обеих сторон. Попытка применения `\gn` в математической формуле приведёт к ошибке, поскольку первый символ `$` в определении `\gn` заставит LaTeX выйти из математического режима. Однако команду `\gn` можно сконструировать так, чтобы её можно было использовать в любом режиме. Эта задача решается при помощи команды `\ensuremath`, которая обеспечивает печать своего аргумента в математическом режиме независимо от текущего режима работы.

Для первичного определения команды `\gn` мы применили `\newcommand`. Для переопределения уже существующей команды необходимо использовать декларацию `\renewcommand`:

```
\renewcommand{\gn}{\ensuremath{\alpha_{v}}}
Пусть \gn\ есть концентрация мух,
причём $\int\gn\,dV=1$.
```

Пусть α_v есть концентрация мух, причём $\int \alpha_v\, dV = 1$.

Она позволяет переопределять даже собственные команды LaTeX'а, а не только введённые пользователем:

```
\Large \renewcommand{\Large}{\tiny}
Большое стало  \Large маленьким!
```

Большое стало маленьким!

Читатель не должен переопределять действие команд LaTeX'а, если плохо представляет себе последствия своих действий, так как изменение одной команды может отразиться на исполнении многих других. Нельзя переопределять команды, чьё имя начинается с `\end`. Имя команды должно начинаться с обратного слеша \ и состоять только из латинских букв, так как наряду с пробелом любая русская буква, цифра, знак препинания, любой из пяти знаков математических операций или любой из десяти служебных символов (раздел 1.4) служат признаком конца имени команды.

LaTeX может не обнаружить ошибку, если имя команды содержит, например, цифру, однако результат будет отличаться от желаемого. Впрочем, допускаются команды с именами, состоящими только из обратного слеша и одного знака

препинания или одной цифры. Все возможные команды с именами, состоящими из обратного слеша и одного знака препинания, знака математической операции или служебного символа, уже заняты. Переопределять же их крайне рискованно. Что касается команд с именами `\0`, `\1`, `\2` ... `\9`, то для официальных руководств по LaTeX'у они просто как бы не существуют. Это значит, что «фирма» не несет ответственности за их применение.

Нужно также предостеречь от определения команд, замещающих командные скобки `\begin` и `\end`. Многие пользователи привычно вводят сокращения наподобие `\be`, `\ee` для ускорения набора уравнений:

```
\newcommand{\be}{\begin{equation}}
\newcommand{\ee}{\end{equation}}
```

В подобных случаях результат не всегда гарантирован. Можно указать несколько случаев, когда подобные определения по меньшей мере бесполезны. Например, команда `\ev` для замены `\end{verbatim}` заведомо вызовет ошибку, поскольку в теле процедуры `verbatim` не распознается ни одна команда, кроме `\end{verbatim}` (раздел 5.5). Также не следует замещать командные скобки для процедур `figure` и `table`, которые создают так называемые плавающие объекты (глава 11), если эти объекты должны «уплывать» в конец документа, как бывает при использовании пакета endfloat.

Вновь определяемые команды могут иметь до девяти аргументов. Рассмотрим следующий пример:

```
\newcommand{\F}[2]{#2_{0} \ldots #2_{#1}}
Последовательность $\F{k}{x}$ содержит
$k+1$ член.
```

Последовательность $x_0 \ldots x_k$ содержит $k+1$ член.

Необязательный аргумент 2 в квадратных скобках в определении команды `\F` указывает, что она имеет два аргумента. Обозначения `#1` и `#2` в последнем аргументе `\newcommand` называются параметрами; при исполнении команды `\F` они замещаются соответственно её первым и вторым аргументами.

LaTeX запоминает определение новой команды так, как оно записано в декларации `\newcommand`. Когда LaTeX исполняет команду `\F`, он замещает обозначение `\F{k}{x}` её определением, подставляя аргументы команды `\F` вместо параметров `#1` и `#2`. Затем LaTeX обрабатывает полученный таким способом текст почти так же, как если бы он был набран во входном файле вместо команды `\F`. Исключение составляют пробелы после имени команды — они игнорируются. Определение команд, которые бы добавляли пробелы после себя, вряд ли станет удачным изобретением, так как может привести к появлению пробелов, нежелательных в некоторых ситуациях.

Один аргумент у новоиспечённой команды можно объявить необязательным. Для этого при определении команды необходимо указать его значение по умолчанию. Продолжая упражняться с командой `\F`, переопределим её так, чтобы она имела необязательный аргумент:

```
\renewcommand{\F}[2][k]{#2_0 \ldots #2_{#1}}
Сравните $\F{x}$ и $\F[N]{x}$.
```

Сравните $x_0 \ldots x_k$ и $x_0 \ldots x_N$.

Необязательным является всегда первый аргумент. В определении команды он замещается параметром `#1`.

Фигурные скобки, окружающие последний аргумент деклараций `\newcommand` или `\renewcommand`, не являются частью определения новой команды. Это означает, что эти фигурные скобки не ограничивают область действия деклараций, которые могут входить в определение новой команды. При необходимости область действия таких деклараций должна быть указана дополнительными фигурными скобками:

```
\newcommand{\good}[2]{{#1}($#2$)}
\newcommand{\bad}[2]{#1($#2$)}
Функция \good{\em F}{x} хорошая, а в
\bad{\em G}{y} действие декларации \verb|\em|
распространяется на последующий текст.
```

Функция $F(x)$ хорошая, а в *G(y) действие декларации* `\em` *распространяется на последующий текст.*

Поскольку символы `$` также ограничивают область действия деклараций, запись `${#2}$` была бы избыточной: она полностью эквивалентна `$#2$`.

Подведём итоги. Команды

⚠ `\newcommand{`*cmd*`}[`*num*`][`*opt*`]{`*def*`}`
⚠ `\renewcommand{`*cmd*`}[`*num*`][`*opt*`]{`*def*`}`
⚠ `\providecommand{`*cmd*`}[`*num*`][`*opt*`]{`*def*`}`

являются декларациями. Первые две соответственно определяют новую и переопределяют уже существующую команду *cmd*, а не упоминавшаяся доселе команда `\providecommand` действует, как `\newcommand`, если *cmd* ранее не была определена, либо сохраняет действующее определение, если *cmd* уже существует. Аргументы деклараций имеют следующее назначение:

cmd — имя команды; должно начинаться с `\` и состоять из последовательности латинских букв либо из одной не буквы (т. е. цифры, знака препинания или служебного символа). Для `\newcommand` команда с именем *cmd* не должна быть ранее определена и не должна начинаться с `\end`. Напротив, для `\renewcommand` имя *cmd* должно быть введено ранее;

num — целое число от 1 до 9, означающее количество аргументов у вновь определяемой команды. По умолчанию новая команда не имеет аргументов;

opt — значение необязательного аргумента. При наличии `[`*opt*`]` первый аргумент *cmd* объявляется необязательным и по умолчанию имеет значение *opt*; при отсутствии `[`*opt*`]` все аргументы являются обязательными;

def — текст, который подставляется вместо каждого появления *cmd* во входном файле. Если в *def* находится параметр вида `#`*n*, вместо него подставляется *n*-й аргумент команды *cmd*. Фигурные скобки, окружающие *def*, не ограничивают область действия деклараций, находящихся в *def*. Вновь определяемая

команда является хрупкой, если *def* содержит хрупкие команды; иначе она является устойчивой. Аргумент *def* может содержать декларации, которые определяют другие команды и процедуры, если те не имеют аргументов.

Новые команды можно определять в любом месте входного файла. При этом следует иметь в виду, что область действия деклараций `\newcommand`, `\renewcommand` и `\providecommand` подчиняется общим правилам, изложенным в разделе 2.5. Поэтому команда, определённая внутри какой-нибудь процедуры, затирается при выходе из неё. Кроме того, область, где сохраняется определение команды, можно явно ограничить фигурными скобками. Иногда это очень полезно для локального переопределения отдельных команд. Однако общее правило гласит, что определения новых команд полезно собирать в одном месте, например в преамбуле. Тогда их легко находить и при необходимости модифицировать.

Команда

`\ensuremath{`*arg*`}`

обеспечивает обработку своего аргумента *arg* в математическом режиме вне зависимости от текущего режима; обычно используется при определении команд или процедур, которые одинаковым образом должны работать как в текстовом, так и в математическом режимах.

Добавим ещё пару слов о стратегии командостроения.

Имена команд должны отражать их назначение. Всего через пару недель будет невозможно вспомнить, для чего предназначалась команда с маловразумительным названием `\BM`:

```
\newcommand{\BM}[1]{\mbox{\mathversion{bold}$#1$}}
```

С помощью этой весьма полезной команды полужирный символ $\boldsymbol{\alpha}$ записывается в виде `\BM{\alpha}`. В научных статьях так обычно печатают обозначения векторов. Поэтому лучше было бы переопределить команду `\vec`:

```
\renewcommand{\vec}[1]{\mbox{\mathversion{bold}$#1$}}
```

поскольку её название более чем красноречиво.

Все свои любимые команды Читатель может собрать в одном файле, назвав его, например, `shorts.tex`, и вводить во входной файл командой `\input {shorts}`. Если когда-нибудь Читатель последует этому совету, полезно будет вспомнить, что пакет bm (раздел 6.6) вводит команду `\bm` с названием, похожим на `\BM`, и с примерно той же целью. Небольшое усовершенствование

```
\providecommand{\bm}[1]{\mbox{\mathversion{bold}$#1$}}
\renewcommand{\vec}[1]{\bm{#1}}
```

позволит задействовать всю мощь пакета bm, одновременно заготовив запасной, более простой вариант набора векторов на случай, если пакет не загружен.

Наконец, выполним своё обещание и расскажем, как создать команду, которая бы не игнорировала пробелы после своего имени. Фокусы с пробелами позволяет проделывать команда

`\xspace` (xspace)

реализованная в пакете xspace Дэвида Карлайла (Carlisle, David). Она вставляет пробел, но только тогда, когда за ней не следует знак препинания, одна из фигурных скобок, закрывающая круглая скобка, прямой или обратный слеш. Если команду `\xspace` добавить в определение другой команды, то та, как правило, оставляет за собой пробел только в необходимых случаях:

```
\newcommand{\gb}{Англия\xspace}
\gb расположена на небольшом острове
у побережья Франции. Это неплохое
местечко, \gb, чтобы там жить.
```

Англия расположена на небольшом острове у побережья Франции. Это неплохое местечко, Англия, чтобы там жить.

В результате отпадает необходимость набирать вслед за именем команды `{}` или `\␣`, чтобы оставить пробел в тексте печатного документа. Недостаток же таких команд в том, что они обходятся с пробелами не так, как все другие, а это легко забыть.

7.2. Определение новых процедур

Декларации



`\newenvironment{`*env*`}[`*num*`][`*opt*`]{`*begdef*`}{`*enddef*`}`
`\renewenvironment{`*env*`}[`*num*`][`*opt*`]{`*begdef*`}{`*enddef*`}`

соответственно определяют новые и переопределяют существующие процедуры. Они могут иметь до пяти аргументов:

env — имя процедуры, которое может содержать любую последовательность букв, цифр и символа `*`, но не должно начинаться с `end`. Для `\newenvironment` *env* не должно быть именем, которое уже было введено ранее. Напротив, для `\renewenvironment` оно уже должно быть определено;

num — количество аргументов у вновь определяемой процедуры; должно быть целым числом от 1 до 9. По умолчанию процедура не имеет аргументов;

opt — значение необязательного аргумента. При наличии `[`*opt*`]` первый аргумент процедуры *env* является необязательным и по умолчанию имеет значение *opt*; при отсутствии `[`*opt*`]` все аргументы *env* являются обязательными;

begdef — текст, подставляемый вместо `\begin{`*env*`}`. Стоящий в *begdef* параметр `#`*n* при подстановке заменяется текстом *n*-го аргумента;

enddef — текст, подставляемый вместо `\end{`*env*`}`. Он не может содержать `#`-параметров.

Аргументы *begdef* и *enddef* могут содержать декларации, которые определяют другие команды и процедуры, если те не имеют аргументов.

Фигурные скобки, окружающие аргументы процедуры, определённой посредством `\newenvironment` или `\renewenvironment`, не ограничивают область действия деклараций, содержащихся в аргументах процедуры.

Новая процедура обычно определяется в терминах существующей, такой как `itemize`. Тогда *begdef* содержит команду `\begin{itemize}`, открывающую процедуру `itemize`, а *enddef* — `\end{itemize}`.

```
\newenvironment{emi}[1][\em]
{\begin{itemize} #1}{\end{itemize}}
Это пример процедуры \texttt{emi}.
\begin{emi}
\item Она определена при помощи
      процедуры \texttt{itemize}.
\item Она имеет необязательный параметр.
\end{emi}
```

Это пример процедуры `emi`.

- *Она определена при помощи процедуры `itemize`.*
- *Она имеет необязательный параметр.*

Необязательные аргументы *num* и *opt* позволяют определять процедуры с аргументами. Они работают так же, как в случае с `\newcommand`. Вызовем теперь процедуру `emi`, определённую в предыдущем примере, с необязательным аргументом:

```
Это пример процедуры \texttt{emi}.
\begin{emi}[\scshape]
\item Она определена при помощи
      процедуры \texttt{itemize}.
\item Она имеет необязательный параметр.
\end{emi}
```

Это пример процедуры `emi`.

- ОНА ОПРЕДЕЛЕНА ПРИ ПОМОЩИ ПРОЦЕДУРЫ ITEMIZE.
- ОНА ИМЕЕТ НЕОБЯЗАТЕЛЬНЫЙ ПАРАМЕТР.

Параметры `#1` ... `#9` могут появляться только в первой части определения процедуры, т. е. в *begdef*. Декларация `\newenvironment` проверяет, не была ли процедура с именем *env* определена ранее, помогая тем самым избежать случайного переопределения существующей процедуры. Напротив, при использовании `\renewenvironment` ответственность за последствия целиком ложится на пользователя. Мы вновь предостерегаем Читателя от изменения стандартных процедур LaTeX'а, так как это может привести к неожиданным результатам.

7.3. Теоремы

Книги и статьи по математике обычно содержат теоремы и другие теоремоподобные структуры, такие как леммы, аксиомы и пр. Нематематический текст также может состоять из аналогичных структур: правил, законов, принципов и так далее. Коллекционирование процедур на каждый случай было бы большой роскошью, и поэтому LaTeX располагает декларацией `\newtheorem`, которая позволяет легко конструировать процедуры для каждой конкретной ситуации.

Существуют два варианта `\newtheorem`:

⚠ `\newtheorem{`*env*`}{`*caption*`}[`*within*`]`
⚠ `\newtheorem{`*env*`}[`*theorem*`]{`*caption*`}`

Оба имеют по два обязательных и одному необязательному аргументу:

env — имя процедуры, состоящее из последовательности букв и не совпадающее с именем уже существующей процедуры или счётчика. Для каждой теоремы *env* создаётся счётчик с тем же именем, если в списке аргументов декларации `\newtheorem` отсутствует опция *theorem*. При наличии этой опции для нумерации теоремы используется уже существующий счётчик *theorem* (см. ниже).

caption — текст, который должен быть напечатан в начале процедуры непосредственно перед номером;

within — имя уже существующего счётчика (обычно соответствующего одной из команд секционирования). Если опция *within* присутствует, то счётчик *env* объявляется внутренним для счётчика *within*. Другими словами, он будет автоматически обнуляться при каждом изменении значения *within*, производимым декларацией `\stepcounter{`*within*`}` или `\refstepcounter{`*within*`}` (раздел 2.9);

theorem — имя определённой ранее теоремы. Если опция *theorem* присутствует в списке аргументов декларации `\newtheorem`, то для нумерации процедуры *env* будет использован счётчик *theorem*, а процедуры *theorem* и *env* будут иметь единую нумерацию.

При каждом вызове процедуры *env* соответствующий ей счётчик (*env* или *theorem*) увеличивается на единицу, а команда `\the`*env* (или `\the`*theorem*) становится текущим `ref`-значением и используется для организации перекрёстных ссылок (раздел 3.7). Первоначально команда `\the`*env* определена как `\arabic{`*env*`}` (или как `\the`*within*.`\arabic{`*env*`}`, если процедура определена с опцией *within*). Исходное определение команды `\the`*theorem* зависит от того, как определена процедура *theorem*. Команды `\the`*env* и `\the`*theorem* могут быть переопределены при помощи `\renewcommand`.

Декларация `\newtheorem` является глобальной, так как она работает со счётчиками, а область определения счётчиков не может быть ограничена никакими фигурными или командными скобками (раздел 2.9). Теоремоподобная процедура может иметь один необязательный аргумент:

`\begin{`*env*`}[`*text*`]`

Обычно в качестве *text* указывают имя автора теоремы, аксиомы или закона; *text* будет вставлен в печатный документ после номера теоремы.

Перейдём к примерам. Начнём с самого простого, но сразу продемонстрируем схему организации перекрёстных ссылок:

```
\newtheorem{teo}{Теорема}
Теорема \ref{Пифагор}
принадлежит Пифагору.
\begin{teo}\label{Пифагор}
Квадрат гипотенузы равен сумме
квадратов катетов.   \end{teo}
```

Теорема 1 принадлежит Пифагору.

Теорема 1 *Квадрат гипотенузы равен сумме квадратов катетов.*

Здесь для нумерации теоремы заведён счётчик `teo`. Первоначально он равен нулю, но каждое исполнение одноимённой процедуры увеличивает его значение на единицу. По умолчанию заголовок теоремы имеет полужирную насыщенность, а её формулировка — курсивное начертание.

В следующем примере теорема `ttt` нумеруется независимо в пределах каждой секции, что определено опцией `section` в декларации `\newtheorem`. Другими словами, счётчик `ttt` обнуляется каждой командой `\section`, начинающей заголовок раздела.

```
\newtheorem{ttt}{Лемма}[section]
\begin{ttt}
  Волга впадает в Каспийское море.
\end{ttt}
```

Лемма 7.3.1 *Волга впадает в Каспийское море.*

В следующем примере вводится теоремоподобная процедура `hop`, которая использует нумерацию, единую с процедурой `ttt`.

```
\newtheorem{hop}[teo]{Гипотеза}
\begin{hop}
  Солнце восходит на востоке.
\end{hop}
```

Гипотеза 2 *Солнце восходит на востоке.*

Помимо номера, теоремы могут иметь названия. Название можно указать в опции теоремоподобной процедуры. Например, при помощи введённой ранее процедуры `teo` теорема Ферма формулируется следующим образом:

```
\begin{teo}[Fermat]
Нет целых чисел $n>2$, $x$, $y$
и $z$ таких, что $x^n+y^n=z^n$.
\end{teo}
```

Теорема 3 (Fermat) *Нет целых чисел $n > 2$, x, y и z таких, что $x^n + y^n = z^n$.*

Поскольку `\newtheorem` является глобальной декларацией, её лучше всего помещать в преамбулу. Размещение `\newtheorem` в недрах исходного текста может вызвать совершенно неожиданные затруднения, если входной файл разбит на несколько файлов и они вводятся в корневой файл командой `\include` (раздел 3.8). Точно так же в файле, вводимом командой `\include`, нельзя создавать новые счётчики (раздел 3.8).

7.3.1. Пакет theorem

Пакет theorem Франка Миттельбаха (Mittelbach, Frank) из коллекции tools расширяет средства проектирования новых теоремоподобных процедур, вводя понятие *стиля теоремы*. Стиль теоремы выбирается декларацией

`\theoremstyle{style}` (theorem)

из 6-ти вариантов *style*:

- `plain` полностью воспроизводит оригинальный вариант оформления теорем, принятый в формате LaTeX, но несколько увеличивает вертикальные пробелы между теоремой и окружающим текстом;
- `change` аналогичен стилю `plain`, но переставляет местами номер теоремы и её заголовок;
- `margin` аналогичен стилю `plain`, но выносит номер теоремы на левое поле страницы;
- `break` размещает заголовок теоремы в отдельной строке;
- `changebreak` аналогичен стилю `break`, но переставляет местами номер теоремы и её заголовок;
- `marginbreak` аналогичен стилю `break`, но выносит номер теоремы на поля страницы.

Теорема будет иметь тот стиль, который был выбран на момент её определения посредством `\newtheorem`. Таким образом, в результате следующих определений

```
\theoremstyle{break}  \newtheorem{Cor}{Предположение}
\theoremstyle{plain}  \newtheorem{Exa}{Пример}[section]
```

теорема `Cor` будет форматировать текст в стиле `break`, а теорема `Exa` — в стиле `plain`. Все последующие теоремы (если они есть) также будут иметь стиль `plain`, пока не встретится другая декларация `\theoremstyle`. Область действия `\theoremstyle` можно фиксировать фигурными скобками, тогда как декларация `\newtheorem` действует глобально.

Шрифт, используемый той или иной теоремой по умолчанию, можно изменять независимо от её стиля. Декларации переключения шрифта, вставляемые непосредственно перед печатью основного содержания теоремы, извлекаются из аргумента декларации

`\theorembodyfont{font-dcls}` (theorem)

Продолжая определять новые теоремоподобные процедуры, добавим ещё одну строку к примеру, начатому выше:

```
\theoremstyle{break}  \newtheorem{Cor}{Предположение}
\theoremstyle{plain}  \newtheorem{Exa}{Пример}[section]
{\theorembodyfont{\upshape} \newtheorem{Rem}{Замечание}}
```

Она устанавливает, что содержание всех теорем `Rem` будет напечатано шрифтом `\upshape` при использовании текущего стиля (т. е. стиля `plain`). Как и в случае с `\theoremstyle`, используется текущее содержание `\theorembodyfont` на момент исполнения `\newtheorem`. Так как последняя строка целиком заключена в фигурные скобки, действие `\theorembodyfont` не будет пролонгировано на последующие определения теорем, если такие будут добавлены. По умолчанию (когда `\theorembodyfont` отсутствует или ничего не содержит) используется шрифт, установленный выбранным стилем. Все стили, кроме `plain`, по умолчанию выбирают `\slshape`.

Можно также изменить шрифт заголовков теорем при помощи

`\theoremheaderfont{`*font-dcls*`}` (theorem)

В отличие от `\theorembodyfont` последняя декларация глобальна. Поэтому её следует использовать в преамбуле и только в единственном числе.

Наконец, ещё две команды

`\theorempreskipamount`
`\theorempostskipamount` (theorem)

есть растяжимые длины, которые задают размер вертикальных промежутков соответственно перед и после теоремы.

В разделе 8.13 описан пакет amsthm, который выполняет примерно те же функции, что и theorem по отношению к определению новых теоремоподобных процедур, но ещё вводит процедуру `proof`, весьма удобную для изложения доказательств теорем.

7.4. Пакет ifthen

TeX имеет примитивные, по нынешним меркам, средства программирования. Они доступны во входном файле, предназначенном для LaTeX'а. Однако мы слишком отклонились бы от цели, начни сейчас рассказ о премудростях TeX'а, поскольку синтаксис его команд отличается от того, к чему Читатель, возможно, уже привык, осилив немалую часть нашей книги. К LaTeX'у прилагается пакет ifthen, который вводит простейшие средства программирования в привычном Читателю ключе, в том числе команды проверки условий и организации циклов. Впрочем, если наш уважаемый Читатель никогда в жизни не программировал, он может смело пропустить остаток главы.

Пакет вводит две команды. Первая из них

`\ifthenelse{`*test*`}{`*then-txt*`}{`*else-txt*`}` (ifthen)

имеет три аргумента. Сначала она проверяет условие, содержащееся в первом аргументе *test*. Если условие выполнено, то содержимое второго аргумента *then-txt* обрабатывается так, как если бы на месте команды `\ifthenelse` со всеми её аргументами был только её второй аргумент *then-txt* (без фигурных скобок). Если

условие нарушено, аналогичным образом обрабатывается третий аргумент *else-txt*. Эти два аргумента могут содержать любой текст, команды и процедуры, допустимые в данном месте входного файла. Первый аргумент *test* должен содержать выражение, которое LaTeX может оценить либо как `true` (верно), либо как `false` (неверно). Это выражение может иметь одну из следующих форм:

num_1 *op* num_2 Сравнение чисел num_1 и num_2. Здесь *op* — один из трёх символов `<`, `>`, `=`. Например, соотношение `\value{page}>17` оценивается как `true` (верно), если текущее значение счётчика страниц `page` больше 17.

`\equal{`str_1`}{`str_2`}` Сравнение строк. Выражение оценивается как `true`, если компилятор считает строки str_1 и str_2 одинаковыми. LaTeX может рассматривать строки str_1 и str_2 как разные, даже если они будут напечатаны одинаково. Например, `\today` и `1␣мая␣2003` не равны даже 1 мая 2003 года. LaTeX считает строки str_1 и str_2 одинаковыми, если замещение всех команд их определениями делает строки одинаковыми. Например, строки `\3{C}P` и `CCCP` одинаковы, если команда `\3` определена следующим образом:

```
\newcommand{\3}[1]{#1#1#1}
```

Если заранее не ясно, могут ли быть равны две строки, надо немного поэкспериментировать.

`\lengthtest{`len_1 *op* len_2`}` Сравнение длин len_1 и len_2, причём *op* — один из трёх символов `<`, `>`, `=`. Например, `\lengthtest{\parindent<1cm}` оценивается как `true`, если длина отступа в начале абзаца `\parindent` меньше одного сантиметра. При сравнении растяжимая длина заменяется её естественной длиной.

`\isodd{`*num*`}` Соответствует `true`, если число *num* нечётно. Может использоваться при проверке, является ли текущая страница правой (нечётной) или левой (чётной). Однако очевидное решение `\isodd{\value{page}}` в данном случае не работает, так как LaTeX «выпускает» страницу (и соответственно увеличивает значение счётчика `page`), имея сформатированными «про запас» ещё несколько абзацев. Вместо этого следует нужное место входного файла пометить при помощи `\label{`*key*`}` и использовать `\isodd{\pageref{`*key*`}}`. Заметим также, что счётчик `page` имеет ожидаемое значение при обработке текста, идущего в верхние и нижние колонтитулы страницы (раздел 17.1).

`\boolean{`*bool*`}` Определяет текущее значение булевой переменной *bool*, где *bool* может быть любой последовательностью букв.

Булева переменная должна быть объявлена декларацией

`\newboolean{`*bool*`}` (ifthen)

а её значение устанавливается декларацией

`\setboolean{`*bool*`}{`*true-or-false*`}` (ifthen)

где true-or-false есть либо `true`, либо `false`. В аргументе test допускается также комбинирование перечисленных выше условий. Комбинирование осуществляется логическими операторами

| `\and` (и) `\or` (или) `\not` (не) | (ifthen) |

из простых условий, сгруппированных при помощи команд `\(` и `\)`, исполняющих роль скобок.

Вторая основная команда, вводимая пакетом ifthen,

| `\whiledo{test}{do}` | (ifthen) |

повторяет процесс do до тех пор, пока условие test верно. Она ничего не делает, если условие test изначально не выполняется. Условие составляется так же, как для команды `\ifthenelse`.

Возможности, которые пакет ifthen предоставляет опытному программисту, продемонстрируем на примере создания команды `\Gcd` для вычисления наибольшего общего делителя двух чисел. Она использует два счётчика `ca` и `cb`, последовательно вычитая меньшее число из большего, пока они не станут равны:

```
\newcounter{ca}        % Вводим счётчик ca.
\newcounter{cb}        % Вводим счётчик cb.
\newcommand{\Gcd}[2]{  % Определяем команду с 2-мя аргументами.
  \setcounter{ca}{#1}  % Присваиваем начальные значения.
  \setcounter{cb}{#2}
  Gcd(#1,#2) =%        % Печатаем начальные значения.
    \whiledo{          % Начинаем цикл.
      \not\(\value{ca}=\value{cb}\)%     % Проверяем условие.
    }{%
      \ifthenelse{\value{ca}>\value{cb}} % Выбираем большее значение.
        {\addtocounter{ca}{-\value{cb}}}
        {\addtocounter{cb}{-\value{ca}}}
      gcd(\arabic{ca},\arabic{cb}) =%    % Печатаем текущий результат.
    }%                 % Заканчиваем цикл.
  \arabic{ca}%         % Печатаем итоговый результат.
}                      % Заканчиваем определение команды.
```

Поскольку в определении команды `\Gcd` предусмотрен вывод результата на каждом шаге вычислений, `\Gcd{54}{30}` напечатает всю цепочку преобразований: Gcd(54,30) = gcd(24,30) = gcd(24,6) = gcd(18,6) = gcd(12,6) = gcd(6,6) = 6.

7.5. Пакет calc

Пакет `calc` переопределяет декларации

```
⚠ \setcounter{ctr}{integer-expr}
⚠ \addtocounter{ctr}{integer-expr}
  \setlength{len-cmd}{glue-expr}
  \addtolength{len-cmd}{glue-expr}
```
(calc)

так, что вместо простого числа *num* или длины *len* (разделы 2.9 и 2.10) теперь они могут воспринимать в своих аргументах соответственно арифметические выражения *integer-expr* и растяжимые длины {*glue-expr*}.

Целочисленные арифметические выражения *integer-expr* могут содержать счётчики, целые числа (или команды, в которых хранятся целые числовые значения) и бинарные операции сложения (+), вычитания (-), умножения (*) и деления (/). Круглые скобки используются, чтобы изменить порядок выполнения операций. В операциях с целыми числами дробная часть отбрасывается.

```
\newcounter{leaf}
\setcounter{leaf}{(\value{page}+1)/2}
Пока напечатано \thepage\ страниц
на \theleaf\ листах.
```

Пока напечатано 170 страниц на 85 листах.

При операциях с длинами все части арифметического выражения *glue-expr*, которые складываются или вычитаются, должны быть одного типа. Например, нельзя писать `2cm+3`, но выражение `2cm+3pt` является правильным. Нельзя также делить на длину, равно как и умножать на длину, то есть в операциях умножения и деления длина всегда должна быть первым сомножителем или делимым. Например, ошибкой будет `4*2cm`, `2cm*4pt`; правильно: `2cm*4`. Интересно, что при умножении растяжимой длины на целое число происходит увеличение как естественной, так и растяжимой части длины. Например, после

```
\setlength{\parskip}{3pt plus 3pt * 2}
```

вертикальный пробел `\parskip` между абзацами будет иметь значение `6pt plus 6pt`. Напомним, что слова `plus` и `minus` характеризуют степень растяжимости длины (раздел 2.10); их не следует пытать со знаками математических операций `+` и `-`.

Ограничение по делению и умножению только на целое число снимается при помощи команд

```
\real{decimal-num}
\ratio{len-expr1}{len-expr2}
```
(calc)

Первая из них просто конвертирует текст *decimal-num* из цифр и десятичной точки в вещественное число, а вторая обозначает число, получаемое от деления *len-expr1* на *len-expr2*.

При умножении растяжимой длины на вещественное число растяжимость пропадает. Так что в результате

```
\setlength{\parskip}{3pt plus 3pt * \real{1.5}}
```

вертикальный пробел `\parskip` между абзацами будет всегда равен 4,5 pt, а не `4.5pt plus 4.5pt`, как можно было бы предположить по аналогии с умножением растяжимой длины на целое число.

В операциях с любыми числами LaTeX отбрасывает дробную часть, если результат должен быть целым числом. Например,

```
\setcounter{x}{7/2}
\setcounter{y}{3*\real{1.6}}
\setcounter{z}{3*\real{1.7}}
```

приписывает значение 3 счётчику `x`, значение 4 — счётчику `y` и значение 5 — счётчику `z`. Такое округление вниз применяется также ко всем промежуточным результатам вычислений, поэтому следующая команда

```
\setcounter{x}{3 * \real{1.6} * \real{1.7}}
```

приравняет `x` к 6.

Поскольку большинство команд LaTeX'а проводят манипуляции с длинами при помощи `\setlength` и `\addtolength`, в их аргументах, принимающих параметры длины, после загрузки пакета calc также разрешается использовать арифметические операции и растяжимые длины. С несколькими такими командами и процедурами мы познакомимся в главе 9. А в следующей главе у нас вновь урок математики.

Нет ничего невозможного для того,
кто не обязан делать это сам.
Закон Вейлера

Глава 8

AMS-LaTeX

В то самое время, когда Лэсли Лампорт (Lamport, Leslie) разрабатывал LaTeX, Майкл Спивак (Spivak, Michael) по заказу Американского математического общества (AMS) работал над созданием формата, известного ныне как AMS-TeX [6]. AMS-TeX значительно расширил средства форматирования математических выражений, но в остальном застыл на уровне Plain TeX'а. Например, AMS-TeX не имел встроенных средств автоматической нумерации и перекрёстного цитирования. Позднее Американское математическое общество решило объединить достоинства AMS-TeX'а и LaTeX'а, выпустив в 1990 году формат AMS-LaTeX. Работу по сращиванию двух форматов выполнили Франк Миттельбах (Mittelbach, Frank) и Райнер Шопф (Schöpf, Rainer) при содействии Майкла Доунса (Downes, Michael) из технического руководства AMS. После появления LaTeX 2_ε AMS-LaTeX был разбит на две коллекции пакетов: AMSFonts и собственно AMS-LaTeX[1].

Пакеты из коллекции AMSFonts вводят множество новых команд для набора огромного количества дополнительных математических символов, имеющихся в шрифтах Euler, разработанных Германом Цапфом (Zapf, Hermann) по заказу AMS. В этой коллекции головным является пакет amssymb. Именно он содержит определения большей части новых команд.

Напомним, что пакеты загружает команда \usepackage. Чтобы приобщиться к AMS-LaTeX'у и получить доступ к дополнительным математическим символам, в преамбулу входного файла достаточно вставить команду

```
\usepackage{amssymb,amsmath}
```

Пакеты могут иметь опции. Они перечислены в разделах 8.2.1 и 8.5.1.

В коллекции AMS-LaTeX головным является пакет amsmath. Он наследует основные средства форматирования математических формул, разработанные Майклом Спиваком, загружая другие пакеты коллекции.

8.1. Кому нужен AMS-LaTeX?

Если предназначение пакетов коллекции AMSFonts примерно понятно, то зачем нужны пакеты AMS-LaTeX заслуживает отдельного обсуждения. Если нашему

[1] Мы описываем версию 2.2f пакетов AMSFonts и версию 2.0 пакетов AMS-LaTeX.

Читателю приходится готовить тексты с изрядной порцией математики, и если однажды он почувствует, что недюжинные математические познания LaTeX'а его уже не удовлетворяют, он по достоинству оценит дополнительные возможности, которые предоставляет AMS-LaTeX. Вот их краткий перечень.

- AMS-LaTeX более аккуратен при форматировании длинных формул. Он расширяет набор процедур для форматирования длинных формул и систем уравнений, записываемых в несколько строк. Эти процедуры автоматически смещают номера уравнений вверх или вниз, чтобы они не печатались поверх уравнений (как случается при использовании процедуры `eqnarray`).

- AMS-LaTeX предоставляет простой способ определения новых обозначений функций типа ln и sin, обеспечивающий корректную расстановку пробелов и автоматический подбор размера шрифта при использовании этих обозначений в индексах.

- AMS-LaTeX вводит удобный способ набора многострочных индексов (например, для указания пределов многомерного суммирования).

- AMS-LaTeX упрощает замещение автоматически вырабатываемого номера выбранного уравнения другим номером или меткой.

- AMS-LaTeX позволяет использовать независимую нумерацию внутри системы уравнений.

- AMS-LaTeX вводит команду `\boldsymbol` для набора полужирным шрифтом отдельных символов, включая такие как ∞, α, β и т. д.

8.2. Коллекция пакетов AMSFonts

Коллекция пакетов AMSFonts поставляется с дополнительными шрифтами, предназначенными для набора математических символов. Часть этих шрифтов восполняет некоторые пробелы в шрифтах семейства Computer Modern, которые используются в документах LaTeX по умолчанию. Главным образом, это шрифты малых кеглей, которые иначе подменяются уменьшенными копиями шрифтов, спроектированных для кегля 10. Изменения от подобного «восполнения» заметны только специалистам. Другая часть содержит шрифты Euler, названные их создателем Германом Цапфом (Zapf, Hermann) в честь знаменитого математика Леонарда Эйлера (Euler, Leonhard). Отличие Эйлеровых шрифтов от Computer Modern видно, что называется, невооружённым глазом. Например, контурные или готические символы в семействе Computer Modern вообще отсутствуют, а набор рукописных букв в Эйлеровых шрифтах заметно богаче.

Подключение добавленных шрифтов производится путём загрузки пакетов, которые входят в состав коллекции AMSFonts:

amsfonts проводит необходимые операции для загрузки Эйлеровых шрифтов, а также вводит математические алфавиты \mathbb и \mathfrak, которые используют эти шрифты;

amssymb вводит полный комплект команд для печати всех доступных математических символов; автоматически загружает пакет amsfonts;

cmmib57 используется для работы с PostScript версией Эйлеровых шрифтов. В этой версии пропущены шрифты некоторых размеров;

eucal вводит математический алфавит \mathscr (или \mathcal в зависимости от выбора опций пакета) для печати каллиграфических букв;

eufrak вводит математический алфавит \mathfrak для печати готических букв; избыточен при использовании пакета amsfonts.

8.2.1. Опции пакетов AMSFonts

Пакеты из коллекции AMSFonts имеют одну общую опцию psamsfonts. Её следует использовать для работы с PostScript версией Эйлеровых шрифтов. Эти шрифты не имеют кеглей 6, 8 и 9, и опция psamsfonts указывает, что LaTeX не должен использовать шрифты соответствующего размера, замещая их имеющимися. Как сказано в документации к пакетам AMSFonts, опцию psamsfonts необходимо использовать, если при компиляции документа появляется сообщение, что не найден метрический файл какого-либо шрифта, например:

```
! Font \U/AMSa/m/n/9=msam9 not loadable: Metric (TFM) file not found.
```

Однако на практике подобное развитие событий совершенно немыслимо, если используется какая-либо современная реализация системы LaTeX, поскольку отсутствующий файл будет сгенерирован «на лету» из шрифта METAFONT. В главе 16 мы расскажем о том, что такое шрифты PostScript и METAFONT, а здесь ограничимся простым рецептом. Если Читатель знает, что использует шрифты PostScript, то он должен загружать пакеты коллекции AMSFonts с опцией psamsfonts. Например:

```
\usepackage[psamsfonts]{amssymb}
```

Можно также добавить psamsfonts в список опций \documentclass — там она будет воспринята сразу всеми пакетами. Мы рекомендуем использовать опцию psamsfonts постоянно. При использовании шрифтов METAFONT это приведёт к микроскопической деградации качества символов некоторых размеров при просмотре документа в DVI-обозревателе. Зато если установлена PostScript-версия шрифтов, качество документов PDF будет отменным[2].

Пакет eucal дополнительно распознаёт опции mathcal и mathscr, имеющие противоположное действие; по умолчанию используется mathcal (раздел 8.3.3).

[2] В MiKTeX'е эти шрифты устанавливаются одновременно с установкой AMSFonts, хотя формально не считаются частью этой коллекции.

Таблица 8.1

Готический алфавит (пакеты amsfonts, eufrak)

				\mathfrak				
$\mathfrak{A}\mathfrak{a}$	$\mathfrak{B}\mathfrak{b}$	$\mathfrak{C}\mathfrak{c}$	$\mathfrak{D}\mathfrak{d}$	$\mathfrak{E}\mathfrak{e}$	$\mathfrak{F}\mathfrak{f}$	$\mathfrak{G}\mathfrak{g}$	$\mathfrak{H}\mathfrak{h}$	$\mathfrak{I}\mathfrak{i}$
$\mathfrak{J}\mathfrak{j}$	$\mathfrak{K}\mathfrak{k}$	$\mathfrak{L}\mathfrak{l}$	$\mathfrak{M}\mathfrak{m}$	$\mathfrak{N}\mathfrak{n}$	$\mathfrak{O}\mathfrak{o}$	$\mathfrak{P}\mathfrak{p}$	$\mathfrak{Q}\mathfrak{q}$	$\mathfrak{R}\mathfrak{r}$
$\mathfrak{S}\mathfrak{s}$	$\mathfrak{T}\mathfrak{t}$	$\mathfrak{U}\mathfrak{u}$	$\mathfrak{V}\mathfrak{v}$	$\mathfrak{W}\mathfrak{w}$	$\mathfrak{X}\mathfrak{x}$	$\mathfrak{Y}\mathfrak{y}$	$\mathfrak{Z}\mathfrak{z}$	$\mathfrak{0}$
$\mathfrak{1}$	$\mathfrak{2}$	$\mathfrak{3}$	$\mathfrak{4}$	$\mathfrak{5}$	$\mathfrak{6}$	$\mathfrak{7}$	$\mathfrak{8}$	$\mathfrak{9}$

8.3. Математические алфавиты

8.3.1. Контурные буквы

Пакет amsfonts определяет команду

`\mathbb{cap_lett}` (amsfonts)

которая по терминологии, принятой в LaTeX 2ε, является математическим алфавитом (раздел 6.6). Она печатает контуры прописных букв:

```
$\mathbb{TOPAZ} \neq \mathcal{TOPAZ}$
```
$\mathbb{TOPAZ} \neq \mathcal{TOPAZ}$

В этом алфавите содержатся только заглавные (прописные) латинские буквы; нет строчных букв, цифр или иных символов.

8.3.2. Готические буквы

Команда

`\mathfrak{lett}` (eufrak, amsfonts)

печатает латинские буквы (прописные и строчные) готическим шрифтом.

```
$\mathfrak{Deutsch} \neq \mathcal{D}eutsch$
```
$\mathfrak{Deutsch} \neq \mathcal{D}eutsch$

Табл. 8.1 содержит полный перечень имеющихся готических символов. Если из всех возможностей, предоставляемых пакетом amsfonts, нужен только готический шрифт, вместо amsfonts можно загрузить пакет eufrak.

8.3.3. Каллиграфические буквы

Пакет eucal изменяет команду LaTeX'а `\mathcal`. Напомним, что она печатает каллиграфические прописные латинские буквы в математических формулах (раздел 6.6). При загрузке пакета eucal эти буквы печатаются шрифтом Euler, а не Computer Modern. Однако если пакет eucal загрузить с опцией `mathscr`, то действие команды `\mathcal` не изменится, а для каллиграфических букв Euler будет введена команда

`\mathscr{capital letters}` (eucal)

Таблица 8.2

Каллиграфические буквы (пакет eucal)

\mathscr									\mathcal								
$\mathscr{A}$	$\mathscr{B}$	$\mathscr{C}$	$\mathscr{D}$	$\mathscr{E}$	$\mathscr{F}$	$\mathscr{G}$	$\mathscr{H}$	$\mathscr{I}$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$	$\mathcal{H}$	$\mathcal{I}$
$\mathscr{J}$	$\mathscr{K}$	$\mathscr{L}$	$\mathscr{M}$	$\mathscr{N}$	$\mathscr{O}$	$\mathscr{P}$	$\mathscr{Q}$	$\mathscr{R}$	$\mathcal{J}$	$\mathcal{K}$	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{N}$	$\mathcal{O}$	$\mathcal{P}$	$\mathcal{Q}$	$\mathcal{R}$
$\mathscr{S}$	$\mathscr{T}$	$\mathscr{U}$	$\mathscr{V}$	$\mathscr{W}$	$\mathscr{X}$	$\mathscr{Y}$	$\mathscr{Z}$		$\mathcal{S}$	$\mathcal{T}$	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{W}$	$\mathcal{X}$	$\mathcal{Y}$	$\mathcal{Z}$	

Мы так и сделали, чтобы продемонстрировать различия между двумя видами каллиграфических букв:

```
\usepackage[mathscr]{eucal}
 ...
$\mathscr{TOPAZ} \neq \mathcal{TOPAZ}$
```

...
$\mathscr{TOPAZ} \neq \mathcal{TOPAZ}$

Табл. 8.2 содержит полный перечень каллиграфических символов.

8.3.4. Как обращаться с алфавитами

Команды, обращающиеся к математическим алфавитам, имеют длинные имена: `\mathbf`, `\mathcal`, `\mathfrak` и т. п. При частом использовании этих команд возникает естественное желание сократить их имена. Например, `\mathbf` сократить до `\mb` при помощи

```
\newcommand{\mb}[1]{\mathbf{#1}}
```

Однако подобных переобозначений рекомендуется избегать. Мы уже писали об этом в разделе 7.1. Назначение команд с короткими именами легко забыть. Лучше вводить команды для обозначения конкретных математических объектов. Например, если Читатель желает выделить векторы полужирным шрифтом, можно переопределить имеющуюся команду `\vec{`symb`}`, которая в исходном определении рисует стрелку над symb:

```
\renewcommand{\vec}[1]{\mathbf{#1}}
```

Разумеется, при таком решении следует быть уверенным в том, что в том же печатном документе не придётся пользоваться командой `\vec` в её исходном виде. Теперь можно набрать `\vec{a} + \vec{b}`, чтобы получить $\mathbf{a} + \mathbf{b}$.

Если через несколько месяцев Читатель решит использовать полужирный шрифт для другой цели, а векторы обозначать маленькой стрелкой над буквой, он сможет легко изменить определение команды `\vec`.

Полезно бывает также приписать отдельным буквам, набранным выбранным шрифтом, собственные команды. Например, если приходится использовать обозначения в виде контурных букв $\mathbb{R}$ и $\mathbb{Z}$, можно ввести команды `\R` и `\Z`:

```
\usepackage{amsfonts}
\newcommand{\R}{\mathbb{R}}
\newcommand{\Z}{\mathbb{Z}}
```

Здесь предполагается, что определения новых команд помещены в преамбулу (где им самое место), а пакет amsfonts загружен для того, чтобы иметь возможность использовать математический алфавит `\mathbb`. Если эти команды используются, к примеру, для обозначения комплексных чисел, то такой метод ввода обозначений может быть более удобен, чем специальная команда `\cnum` для обозначения комплексных чисел вообще:

```
\newcommand{\cnum}[1]{\mathbb{#1}}
```

и последующего обращения к ней `\cnum{R}` и `\cnum{Z}` в тексте входного файла (хотя такой способ хорош тем, что напоминает смысл обозначений).

8.4. Пакет amssymb

Шрифты Euler содержат большое количество символов, не имеющих аналогов в шрифтах Computer Modern. Команды для обращения к этим символам определены в пакете amssymb; они перечислены в табл. 8.3–8.10. Для загрузки пакета в преамбулу входного файла необходимо поместить команду

```
\usepackage{amssymb}
```

Теперь любой символ, будь то $\blacktriangle$ (`\blacktriangle`) или $\nsubseteq$ (`\nsubseteq`), имеющийся в таблицах, вводится в печатный документ при помощи соответствующей ему команды. Напомним, что способ позиционирования символа зависит от того, к какому классу он принадлежит (раздел 6.3). Мы сгруппировали математические символы в таблицах по их принадлежности к разным классам.

Пакет amssymb автоматически загружает пакет amsfonts и, следовательно, обладает всеми его возможностями. Явная загрузка обоих пакетов при помощи `\usepackage` не приведёт к каким-либо неприятным последствиям, так как LaTeX проверяет, чтобы любой пакет не был загружен дважды.

Таблица 8.3

Греческие буквы (пакет amssymb)

$\digamma$	`\digamma`	$\varkappa$	`\varkappa`		
$\varGamma$	`\varGamma`	$\varDelta$	`\varDelta`	$\varTheta$	`\varTheta`
$\varLambda$	`\varLambda`	$\varXi$	`\varXi`	$\varPi$	`\varPi`
$\varSigma$	`\varSigma`	$\varUpsilon$	`\varUpsilon`	$\varPhi$	`\varPhi`
$\varPsi$	`\varPsi`	$\varOmega$	`\varOmega`		

Таблица 8.4

Древнееврейские буквы (пакет amssymb)

$\beth$	`\beth`	$\daleth$	`\daleth`	$\gimel$	`\gimel`

Таблица 8.5

Дополнительные символы (пакет amssymb)

$\hbar$	`\hbar`	$\hslash$	`\hslash`		
$\angle$	`\angle`	$\sphericalangle$	`\sphericalangle`	$\measuredangle$	`\measuredangle`
$\nexists$	`\nexists`	$\mho$	`\mho`	$\Finv$	`\Finv`
$\Game$	`\Game`	$\Bbbk$	`\Bbbk`	$\backprime$	`\backprime`
$\circledS$	`\circledS`	$\complement$	`\complement`	$\eth$	`\eth`
$\square$	`\square`	$\blacksquare$	`\blacksquare`	$\varnothing$	`\varnothing`
$\triangledown$	`\triangledown`	$\blacktriangledown$	`\blacktriangledown`	$\blacktriangle$	`\blacktriangle`
$\lozenge$	`\lozenge`	$\blacklozenge$	`\blacklozenge`	$\bigstar$	`\bigstar`
$\diagup$	`\diagup`	$\diagdown$	`\diagdown`		

Таблица 8.6

Разделители (пакет amssymb)

$\ulcorner$	`\ulcorner`	$\urcorner$	`\urcorner`	$\llcorner$	`\llcorner`	$\lrcorner$	`\lrcorner`

Таблица 8.7

Символы бинарных операций (пакет amssymb)

$\ltimes$	\ltimes	$\rtimes$	\rtimes	$\smallsetminus$	\smallsetminus
$\leftthreetimes$	\leftthreetimes	$\rightthreetimes$	\rightthreetimes	$\divideontimes$	\divideontimes
$\lessdot$	\lessdot	$\gtrdot$	\gtrdot	$\dotplus$	\dotplus
$\curlywedge$	\curlywedge	$\curlyvee$	\curlyvee	$\veebar$	\veebar
$\barwedge$	\barwedge	$\doublebarwedge$	\doublebarwedge		
$\circleddash$	\circleddash	$\circledast$	\circledast	$\circledcirc$	\circledcirc
$\Cap$	\Cap	$\Cup$	\Cup	$\intercal$	\intercal
$\boxdot$	\boxdot	$\boxtimes$	\boxtimes	$\centerdot$	\centerdot
$\boxminus$	\boxminus	$\boxplus$	\boxplus		

Таблица 8.8

Символы сравнения (пакет amssymb)

$\leqq$	\leqq	$\geqq$	\geqq	$\backsim$	\backsim
$\leqslant$	\leqslant	$\geqslant$	\geqslant	$\backsimeq$	\backsimeq
$\lesssim$	\lesssim	$\gtrsim$	\gtrsim	$\eqsim$	\eqsim
$\lessapprox$	\lessapprox	$\gtrapprox$	\gtrapprox	$\approxeq$	\approxeq
$\lesseqgtr$	\lesseqgtr	$\gtreqless$	\gtreqless	$\thicksim$	\thicksim
$\lesseqqgtr$	\lesseqqgtr	$\gtreqqless$	\gtreqqless	$\thickapprox$	\thickapprox
$\lessgtr$	\lessgtr	$\gtrless$	\gtrless	$\shortmid$	\shortmid
$\lll$	\lll, \llless	$\ggg$	\ggg, \gggtr	$\shortparallel$	\shortparallel
$\eqslantless$	\eqslantless	$\eqslantgtr$	\eqslantgtr	$\between$	\between
$\risingdotseq$	\risingdotseq	$\fallingdotseq$	\fallingdotseq	$\doteqdot$	\doteqdot
$\Subset$	\Subset	$\Supset$	\Supset	$\bumpeq$	\bumpeq
$\subseteqq$	\subseteqq	$\supseteqq$	\supseteqq	$\Bumpeq$	\Bumpeq
$\sqsubset$	\sqsubset	$\sqsupset$	\sqsupset	$\varpropto$	\varpropto
$\precsim$	\precsim	$\succsim$	\succsim	$\eqcirc$	\eqcirc
$\precapprox$	\precapprox	$\succapprox$	\succapprox	$\circeq$	\circeq
$\preccurlyeq$	\preccurlyeq	$\succcurlyeq$	\succcurlyeq	$\pitchfork$	\pitchfork
$\curlyeqprec$	\curlyeqprec	$\curlyeqsucc$	\curlyeqsucc	$\backepsilon$	\backepsilon
$\trianglelefteq$	\trianglelefteq	$\trianglerighteq$	\trianglerighteq	$\triangleq$	\triangleq
$\vartriangleleft$	\vartriangleleft	$\vartriangleright$	\vartriangleright	$\vartriangle$	\vartriangle
$\blacktriangleleft$	\blacktriangleleft	$\blacktriangleright$	\blacktriangleright	$\therefore$	\therefore
$\smallsmile$	\smallsmile	$\smallfrown$	\smallfrown	$\because$	\because
$\vDash$	\vDash	$\Vvdash$	\Vvdash	$\Vdash$	\Vdash

Таблица 8.9

Символы сравнения (пакет amssymb)

$\nless$	`\nless`	$\ngtr$	`\ngtr`	$\nsim$	`\nsim`
$\nleq$	`\nleq`	$\gneq$	`\gneq`	$\nmid$	`\nmid`
$\nleqslant$	`\nleqslant`	$\ngeqslant$	`\ngeqslant`	$\nshortmid$	`\nshortmid`
$\nleqq$	`\nleqq`	$\ngeqq$	`\ngeqq`	$\nparallel$	`\nparallel`
$\lnsim$	`\lnsim`	$\gnsim$	`\gnsim`	$\nshortparallel$	`\nshortparallel`
$\lnapprox$	`\lnapprox`	$\gnapprox$	`\gnapprox`	$\ncong$	`\ncong`
$\lneq$	`\lneq`	$\ngeq$	`\ngeq`	$\nvdash$	`\nvdash`
$\lneqq$	`\lneqq`	$\gneqq$	`\gneqq`	$\nVdash$	`\nVdash`
$\nprec$	`\nprec`	$\nsucc$	`\nsucc`	$\nvDash$	`\nvDash`
$\precnsim$	`\precnsim`	$\succnsim$	`\succnsim`	$\nVDash$	`\nVDash`
$\npreceq$	`\npreceq`	$\nsucceq$	`\nsucceq`		
$\precneqq$	`\precneqq`	$\succneqq$	`\succneqq`		
$\precnapprox$	`\precnapprox`	$\succnapprox$	`\succnapprox`		
$\ntriangleleft$	`\ntriangleleft`	$\ntriangleright$	`\ntriangleright`		
$\ntrianglelefteq$	`\ntrianglelefteq`	$\ntrianglerighteq$	`\ntrianglerighteq`		
$\lvertneqq$	`\lvertneqq`	$\gvertneqq$	`\gvertneqq`		
$\nsubseteq$	`\nsubseteq`	$\nsupseteq$	`\nsupseteq`		
$\nsubseteqq$	`\nsubseteqq`	$\nsupseteqq$	`\nsupseteqq`		
$\subsetneq$	`\subsetneq`	$\supsetneq$	`\supsetneq`		
$\subsetneqq$	`\subsetneqq`	$\supsetneqq$	`\supsetneqq`		
$\varsubsetneq$	`\varsubsetneq`	$\varsupsetneq$	`\varsupsetneq`		
$\varsubsetneqq$	`\varsubsetneqq`	$\varsupsetneqq$	`\varsupsetneqq`		

Таблица 8.10

Стрелки (пакет amssymb)

$\twoheadleftarrow$	`\twoheadleftarrow`	$\twoheadrightarrow$	`\twoheadrightarrow`	$\leftrightarrow$	`\leftrightarrow`
$\looparrowleft$	`\looparrowleft`	$\looparrowright$	`\looparrowright`	$\multimap$	`\multimap`
$\leftrightharpoons$	`\leftrightharpoons`	$\rightleftharpoons$	`\rightleftharpoons`	$\rightsquigarrow$	`\rightsquigarrow`
$\curvearrowleft$	`\curvearrowleft`	$\curvearrowright$	`\curvearrowright`	$\leftrightsquigarrow$	`\leftrightsquigarrow`
$\upharpoonleft$	`\upharpoonleft`	$\upharpoonright$	`\upharpoonright`	$\upuparrows$	`\upuparrows`
$\downharpoonleft$	`\downharpoonleft`	$\downharpoonright$	`\downharpoonright`	$\downdownarrows$	`\downdownarrows`
$\leftleftarrows$	`\leftleftarrows`	$\rightrightarrows$	`\rightrightarrows`	$\rightleftarrows$	`\rightleftarrows`
$\Lsh$	`\Lsh`	$\Rsh$	`\Rsh`	$\leftrightarrows$	`\leftrightarrows`
$\circlearrowleft$	`\circlearrowleft`	$\circlearrowright$	`\circlearrowright`		
$\leftarrowtail$	`\leftarrowtail`	$\rightarrowtail$	`\rightarrowtail`		
$\dashrightarrow$	`\dashrightarrow`	$\dashleftarrow$	`\dashleftarrow`		
$\Lleftarrow$	`\Lleftarrow`	$\Rrightarrow$	`\Rrightarrow`		
$\nleftarrow$	`\nleftarrow`	$\nrightarrow$	`\nrightarrow`	$\nleftrightarrow$	`\nleftrightarrow`
$\nLeftarrow$	`\nLeftarrow`	$\nRightarrow$	`\nRightarrow`	$\nLeftrightarrow$	`\nLeftrightarrow`

8.5. Коллекция пакетов $\mathcal{AMS}$-LaTeX

Коллекция $\mathcal{AMS}$-LaTeX содержит следующие пакеты:

amsmath вводит дополнительные процедуры для набора многострочных выключных уравнений, а также ряд других средств форматирования математических выражений; amsmath загружает пакеты amstext, amsbsy, amsopn (см. ниже);

amsbsy вводит команды `\boldsymbol` и `\pmb` для набора полужирных символов; автоматически загружается пакетом amsmath;

amscd вводит процедуру `CD` для форматирования коммутативных диаграмм;

amsopn вводит декларацию `\DeclareMathOperator` для определения новых команд, печатающих обозначения функций типа sin и lim; автоматически загружается пакетом amsmath;

amstext вводит команду `\text` для набора фрагмента текста внутри выключных формул; автоматически загружается пакетом amsmath;

amsthm вводит процедуру `proof` (доказательство) и расширенный вариант декларации `\newtheorem`;

amsxtra вводит декларацию `\accentedsymbol` для определения команд, печатающих символы с диакритическими знаками, а также ряд других устаревших команд; поддерживается для совместимости со старыми версиями пользовательских печатных документов;

upref переопределяет команду `\ref` так, чтобы она печатала номера перекрёстных ссылок всегда прямым романским шрифтом независимо от контекста.

Помимо пакетов коллекция $\mathcal{AMS}$-LaTeX содержит классы печатных документов `amsart`, `amsbook` и `amsproc` для оформления статей, книг и докладов в соответствии со стандартами, принятыми в американских математических журналах.

8.5.1. Опции пакета amsmath

Пакет amsmath имеет следующие опции:

`centertags` | `tbtags` устанавливают размещение номеров уравнений, состоящих из нескольких строк. Опция `centertags`, используемая по умолчанию, указывает, что номер уравнения размещается на половине суммарной высоты всех строк уравнения. При наличии опции `tbtags` номер уравнения размещается напротив последней строки уравнения, если уравнения нумеруются справа, или напротив первой строки, если уравнения нумеруются слева (опция `leqno`, см. ниже);

`sumlimits` | `nosumlimits` контролируют размещение нижних и верхних индексов у знака суммы $\sum$ и других символов переменного размера ($\prod$, $\coprod$, $\bigotimes$, $\bigoplus$ и т. д.,

кроме знаков интегрирования и обозначений функций). Опция `sumlimits`, действующая по умолчанию, в выключных уравнениях помещает индексы выше и ниже знака суммы. При наличии опции `nosumlimits` индексы помещаются сбоку даже в выключных уравнениях;

`intlimits` | `nointlimits` контролируют размещение индексов у знаков интегрирования $\int$, $\oint$ и т. д. По умолчанию действует опция `nointlimits`, аналогичная `nosumlimits`;

`namelimits` | `nonamelimits` контролируют размещение индексов у обозначений функций det, inf, lim, max и т. д., у которых в выключных уравнениях индексы традиционно располагаются в позиции пределов над или под обозначением функции; распознаются также пакетом amsopn. По умолчанию используется опция `namelimits`, действующая аналогично `sumlimits`.

Пакет amsmath распознаёт также опции `leqno` и `fleqn` стандартных классов печатных документов (раздел 3.2). Для опции `leqno` пакет amsmath вводит альтернативную ей опцию `reqno`, означающую, что уравнения нумеруются с правой стороны. Стандартные классы используют правую нумерацию по умолчанию.

8.6. Выключные уравнения

Пакет amsmath обеспечивает ряд дополнительных структур для набора выключных уравнений в дополнение к базисным процедурам LaTeX'а `equation` и `eqnarray`. Расширенный набор состоит из процедур

`equation`	`equation*`	`align`	`align*`	`split`
`gather`	`gather*`	`flalign`	`flalign*`	
`multline`	`multline*`	`alignat`	`alignat*`	

Хотя стандартная процедура `eqnarray` по-прежнему доступна, вместо неё рекомендуется использовать `align` или `split`.

Все процедуры, за исключением `split`, автоматически нумеруют уравнения, причём для этого они используют один и тот же счётчик `equation`, чтобы обеспечить единую нумерацию всех формул. Автоматическая нумерация формул подавляется при использовании `*`-формы перечисленных выше процедур. Процедура `split`, естественно, не имеет `*`-формы. Она вообще отличается от других процедур форматирования выключных уравнений, так как предназначена для использования только внутри них (раздел 8.6.3). Для более избирательного управления нумерацией формул служат команды

```
\tag{text}    \tag*{text}
\notag
```

Команда `\notag` (действующая аналогично команде ядра LaTeX'а `\nonumber`) подавляет номер на любой строке уравнения. Её, как и `\tag` или `\tag*`, целесообразно помещать в конце или начале соответствующей строки (например, перед

\\). Команда \tag, напротив, пометит соответствующую ей строку уравнения, даже если автоматическая нумерация отключена (т. е. использована *-форма одной из процедур). Текст *text*, содержащийся в аргументе \tag и используемый в качестве номера уравнения, обрабатывается в текстовой моде. Это может быть произвольный текст типа $1'$ или ii. Команда \tag* печатает номер уравнения *text* без добавления круглых скобок вокруг него. Если \tag или \tag* используются для замещения автоматически вырабатываемого номера уравнения (внутри обычной формы процедур), значение счётчика equation не увеличивается. Следующий пример показывает, как можно нумеровать разные варианты одного уравнения, используя механизм перекрёстного цитирования:

```
Два варианта (\ref{eq:difference}),
(\ref{eq:identity}) одного уравнения:
\begin{equation}\label{eq:difference}
a-b=0,
\end{equation}
\begin{equation}\label{eq:identity}
a=b.     \tag{\ref{eq:difference}$'$}
\end{equation}
```

Два варианта (8.1), (8.1′) одного уравнения:

$$a - b = 0, \tag{8.1}$$

$$a = b. \tag{8.1′}$$

Читатель сам может подсчитать, сколько раз нужно обработать входной файл, чтобы все перекрёстные ссылки в этом примере сработали правильно.

Примеры, позволяющие сравнить работу всех процедур форматирования уравнений, собраны в табл. 8.11 (вертикальные линии в примерах указывают границы рабочего поля страницы). Следует обратить внимание на важное отличие в способе расстановки символов &, служащих для разделения колонок в процедурах align, alignat, falign, split, по сравнению с процедурой eqnarray.

```
Сравните:
\begin{eqnarray}
 a_1 &=& b_1+c_1\\
 a_2 &=& b_2+c_2
\end{eqnarray}
и
\begin{align}
  a_1 &=  b_1+c_1\\
  a_2 &=  b_2+c_2
\end{align}
```

Сравните:

$$a_1 \quad = \quad b_1 + c_1 \tag{8.2}$$
$$a_2 \quad = \quad b_2 + c_2 \tag{8.3}$$

и

$$a_1 = b_1 + c_1 \tag{8.4}$$
$$a_2 = b_2 + c_2 \tag{8.5}$$

Обсудим теперь особенности и назначение каждой из процедур форматирования уравнений.

8.6.1. Отдельное уравнение

\begin{equation}	...	\end{equation}
\begin{equation*}	...	\end{equation*}

(amsmath)

Таблица 8.11

Процедуры форматирования выключных уравнений

`\begin{equation*}` `a=b` `\end{equation*}`	$$a = b$$
`\begin{equation}` `a=b` `\end{equation}`	$$a = b \tag{1}$$
`\begin{equation}` `\begin{split}` `a &= b+c-d+{}\\` `  &\quad +e-f=\\` `  &= g+h=\\` `  &= i` `\end{split}` `\end{equation}`	$$\begin{split} a &= b+c-d+{}\\ &\quad +e-f=\\ &= g+h=\\ &= i \end{split} \tag{2}$$
`\begin{multline}` `a+b+c+d+e+f+{}\\` `+i+j+k+l+m+n` `\end{multline}`	$$\begin{multline} a+b+c+d+e+f+{}\\ +i+j+k+l+m+n \end{multline} \tag{3}$$
`\begin{gather}` `a_1=b_1+c_1\\` `a_2=b_2+c_2-d_2+e_2` `\end{gather}`	$$a_1=b_1+c_1 \tag{4}$$ $$a_2=b_2+c_2-d_2+e_2 \tag{5}$$
`\begin{align}` `a_1 &= b_1+c_1\\` `a_2 &= b_2+c_2-d_2+e_2` `\end{align}`	$$a_1 = b_1+c_1 \tag{6}$$ $$a_2 = b_2+c_2-d_2+e_2 \tag{7}$$
`\begin{align}` `a_{11} &= b_{11}&` `  a_{12} &= b_{12}\\` `a_{21} &= b_{21}&` `  a_{22} &= b_{22}+c_{22}` `\end{align}`	$$a_{11} = b_{11} \qquad a_{12} = b_{12} \tag{8}$$ $$a_{21} = b_{21} \qquad a_{22} = b_{22}+c_{22} \tag{9}$$
`\begin{flalign*}` `a_{11} &= b_{11}&` `  a_{12} &= b_{12}\\` `a_{21} &= b_{21}&` `  a_{22} &= b_{22}+c_{22}` `\end{flalign*}`	$$\begin{flalign*} a_{11} &= b_{11} & a_{12} &= b_{12}\\ a_{21} &= b_{21} & a_{22} &= b_{22}+c_{22} \end{flalign*}$$

Процедура `equation` размещает выключное уравнение в одной строке. Она автоматически печатает номер уравнения. Процедура `equation*` делает всё то же самое, но не ставит номер уравнения, если это не сделано при помощи команды `\tag`. В ядре LaTeX'а процедура `equation*` формально отсутствует, но её функции выполняет `displaymath`. Интересно, что `\tag` метит даже уравнения, созданные при помощи скобок `\[` ... `\]`:

```
\[
  E = mc^2 \tag{\textasteriskcentered}
\]
```

$$E = mc^2 \qquad (*)$$

8.6.2. Расщепление уравнения без выравнивания

```
\begin{multline}   ...  \end{multline}
\begin{multline*}  ...  \end{multline*}
```
(amsmath)

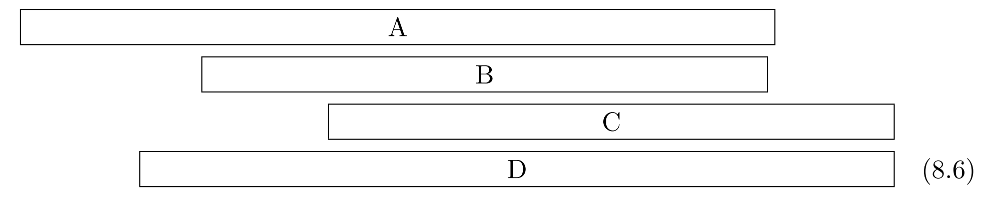
Процедуру `multline` следует использовать вместо `equation`, если уравнение не помещается на одной строке. Точки переноса части уравнения на следующую строку, как обычно, отмечаются командой `\\`. Первая строка уравнения будет сдвинута влево, а последняя — вправо, почти вплотную к полям страницы. Величину отступа от полей определяет командная длина

`\multlinegap` (amsmath)

Строки между первой и последней центрируются (за исключением случая, когда задействована опция `fleqn` в `\documentclass`). Однако их можно сдвигать влево или вправо соответственно командами

`\shoveleft{`*line*`}` `\shoveright{`*line*`}` (amsmath)

В их аргументе должен находиться весь текст, соответствующий сдвигаемой строке уравнения, исключая заключительную команду `\\`. В следующем примере



рамку заданных размеров рисует команда `\framebox`, которую мы изучим в главе 9:

```
\begin{multline}
  \framebox[10cm]{A}\\
  \framebox[7.5cm]{B}\\
  \shoveright{\framebox[7.5cm]{C}}\\
  \framebox[10cm]{D}
\end{multline}
```

Значение `\multlinegap`, как и любой командной длины, можно изменить при помощи `\setlength` и `\addtolength` (раздел 2.10).

8.6.3. Расщепление уравнения с выравниванием

`\begin{split} ... \end{split}`	(amsmath)

Подобно `multline`, процедура `split` предназначена для расщепления на строки отдельных уравнений (но не систем уравнений), не умещающихся в одной строке. В отличие от `multline`, процедура `split` обеспечивает выравнивание расщеплённых строк относительно точек выравнивания, которые, как обычно, устанавливаются амперсантами `&`.

Мы уже отмечали, что процедура `split` не нумерует уравнения, так как предназначена для использования только внутри других процедур форматирования уравнений (обычно `equation`, `align` или `gather`), которые обеспечивают нумерацию. Например, уравнение

$$\begin{split} H_c &= \frac{1}{2n} \sum^n_{l=0}(-1)^{l}(n-{l})^{p-2} \sum_{l_1+\dots+l_p=l}\prod^p_{i=1}\binom{n_i}{l_i} \times \\ &\quad \times [(n-l)-(n_i-l_i)]^{n_i-l_i}\cdot \Bigl[(n-l)^2-\sum^p_{j=1}(n_i-l_i)^2\Bigr] \end{split} \tag{8.7}$$

во входном файле описано следующим образом:

```
\begin{equation}
  \begin{split}
  H_c &= \frac{1}{2n} \sum^n_{l=0}(-1)^{l}(n-{l})^{p-2}
         \sum_{l_1+\dots+l_p=l}\prod^p_{i=1}\binom{n_i}{l_i} \times \\
      &\quad \times [(n-l)-(n_i-l_i)]^{n_i-l_i}\cdot
         \Bigl[(n-l)^2-\sum^p_{j=1}(n_i-l_i)^2\Bigr]
  \end{split}
\end{equation}
```

8.6.4. Системы уравнений без выравнивания

`\begin{gather} ... \end{gather}` `\begin{gather*} ... \end{gather*}`	(amsmath)

Процедура `gather` используется для группы последовательных уравнений, если их не нужно выравнивать. Каждое уравнение центрируется в своей строке независимо от других (см. пример в табл. 8.11).

8.6.5. Системы уравнений с выравниванием

```
\begin{align}      ...  \end{align}
\begin{align*}     ...  \end{align*}
\begin{flalign}    ...  \end{flalign}
\begin{flalign*}   ...  \end{flalign*}
```

(amsmath)

Процедура align используется для двух или более уравнений, которые желательно выровнять по вертикали; обычно выравнивание производится относительно знаков бинарных операций.

Чтобы разместить уравнения в несколько столбцов, можно использовать ту же процедуру align, указав необходимое количество символов & в одной строке:

$$x = y \qquad X = Y \qquad a = b + c \tag{8.8}$$
$$x' = y' \qquad X' = Y' \qquad a' = b \tag{8.9}$$
$$x + x' = y + y' \qquad X + X' = Y + Y' \qquad a'b = c'b \tag{8.10}$$

```
\begin{align}
  x   &=y    & X   &=Y    & a  &=b+c\\
  x'  &=y'   & X'  &=Y'   & a' &=b  \\
  x+x'&=y+y' & X+X'&=Y+Y' & a'b&=c'b
\end{align}
```

Той же цели служит процедура

```
\begin{alignat}{num_col}    ...  \end{alignat}
\begin{alignat*}{num_col}   ...  \end{alignat*}
```

(amsmath)

В ней можно явно указать расстояние между группами уравнений. Она имеет один аргумент *num_col*, который задаёт количество групп уравнений. Максимально допустимое число символов & в строке вычисляется как 2*num_col* − 1 (т. е. один амперсант внутри группы и по амперсанту между группами).

$$C_{11} = C_1 + C \qquad C_{12} = -C \tag{8.11}$$
$$C_{21} = -C \qquad C_{22} = C_2 + C \tag{8.12}$$

```
\begin{alignat}{2}
  C_{11} &= C_1+C   &\qquad  C_{12} &= -C \\
  C_{21} &= -C      &        C_{22} &= C_2+C
\end{alignat}
```

Процедура flalign действует аналогично align, только уравнения прижимаются к полям страницы (см. пример в табл. 8.11).

8.7. Вертикальное позиционирование многострочных уравнений

В отличие от eqnarray процедуры пакета amsmath не допускают автоматической вставки разрыва страницы между строками уравнения, если такая вставка не будет явно разрешена посредством одной из команд

```
\displaybreak[num]
\allowdisplaybreaks[num]
```
(amsmath)

Смысл такого подхода состоит в том, что разрыв страницы посреди уравнения или системы уравнений требует разрешения автора. Ставить `\displaybreak` лучше всего непосредственно перед командой `\\` переноса на следующую строку, где это разрешение может иметь эффект. Подобно `\pagebreak` (раздел 4.7), `\displaybreak` имеет необязательный аргумент *num* — число, которое может изменяться от 0 до 4: при $num = 0$ переход на следующую страницу допустим, но не желателен; команда `\displaybreak` без аргумента эквивалентна `\displaybreak [4]` и делает такой переход обязательным.

Если Читатель предпочитает традиционную для LaTeX'а стратегию, разрешающую разрывать страницу в середине любых многострочных уравнений, ему следует поместить `\allowdisplaybreaks` в преамбулу входного файла. Необязательный аргумент *num*, принимая значения от 1 до 4, может использоваться для более тонкого управления: 1 разрешает разрывы страниц там, где это практически неизбежно; значения 2, 3, 4 означают более высокую степень разрешения. Когда перенос на новую страницу разрешён с помощью `\allowdisplaybreaks`, команда `\\*`, как обычно, может использоваться для запрещения прерывания страницы после текущей строки.

8.8. Текстовые вставки внутри уравнений

Команда

```
\intertext{text}
```
(amsmath)

используется для короткой текстовой вставки в одну-две-три строки внутри многострочных выключных уравнений. Такая вставка сохраняет выравнивание строк, которое было бы нарушено, если просто закончить процедуру и начать новую после текста. Команда `\intertext` может появляться только после команд `\\` или `\\*`. Обратите внимание на положение слов «а также» в следующем примере:

$$A_1 = N_0(\lambda; \Omega') - \phi(\lambda; \Omega'), \tag{8.13}$$

$$A_2 = \phi(\lambda; \Omega') - \phi(\lambda; \Omega), \tag{8.14}$$

а также

$$A_3 = \mathcal{N}(\lambda; \omega). \tag{8.15}$$

```
\begin{align}
  A_1 &= N_0 (\lambda;\Omega') - \phi(\lambda;\Omega'),\\
  A_2 &= \phi(\lambda;\Omega') - \phi(\lambda;\Omega ),\\
\intertext{а также}
  A_3 &= \mathcal{N}(\lambda;\omega).
\end{align}
```

Пояснения внутри математической формулы удобно делать командой

`\text{text}` (amstext)

Она определена в пакете amstext, который автоматически загружается пакетом amsmath. Её основное назначение — вставка в математические формулы слов или коротких фраз, напечатанных тем шрифтом, который использовался непосредственно перед формулой. В этом она напоминает команду `\mbox` (раздел 9.1). Однако в отличие от `\mbox` размер шрифта автоматически уменьшается, если `\text` находится в верхнем или нижнем индексе.

```
...\sffamily Функция
  \begin{equation}
    f_{[x_{i-1},x_i]}
    \text{ монотонна при }
    i=1 \ldots I_{\text{max}}
  \end{equation}
```

...Функция

$$f_{[x_{i-1},x_i]} \text{ монотонна при } i=1\ldots I_{\text{max}}. \quad (8.16)$$

Команды переключения шрифта: `\textrm`, `\textbf`, `\textsl` и т. д. — также не препятствуют уменьшению размера шрифта, когда находятся в индексах, но другие атрибуты шрифта, вообще говоря, не соответствуют действовавшим перед формулой:

```
...\sffamily Функция
  \begin{equation}
    f_{[x_{i-1},x_i]}
    \textsl{ монотонна при }
    i=1 \ldots I_{\textsl{max}}
  \end{equation}
```

...Функция

$$f_{[x_{i-1},x_i]} \textsl{ монотонна при } i=1\ldots I_{\textsl{max}}. \quad (8.17)$$

8.9. Нумерация уравнений

Процедуры пакета amsmath исключают печать номера уравнения поверх его текста, при необходимости перемещая номер на отдельную строку выше или ниже уравнения. Если все-таки положение номера уравнения неудовлетворительно, его можно сместить на некоторое расстояние *len* при помощи команды

`\raisetag{len}` (amsmath)

Например, `\raisetag{6pt}` поднимает последующий номер уравнения на шесть пунктов вверх. Такую корректировку рекомендуется отложить до самой последней стадии редактирования печатного документа.

8.9.1. Иерархия нумераций

Чтобы нумеровать уравнения независимо в пределах каждого раздела, необходимо переопределить команду `\theequation`:

```
\renewcommand{\theequation}{\thesection.\arabic{equation}}
```

Тогда в первом разделе уравнения будут иметь номера (1.1), (1.2) т. д. Однако счётчик уравнений `equation` не обнуляется в начале нового раздела, если этого не сделать явно (используя команду `\setcounter`). Пакет amsmath вводит декларацию

`\numberwithin{`*cnt*`}{`*outcnt*`}` (amsmath)

которая все эти заботы переложит на LaTeX. Чтобы нумеровать уравнения независимо внутри каждого раздела, начинающегося с команды `\section`, достаточно в преамбулу входного файла вписать

```
\numberwithin{equation}{section}
```

Декларацию `\numberwithin` можно применять к любому счётчику, а не только к `equation`.

8.9.2. Перекрёстные ссылки к номерам уравнений

Команда

`\eqref{`*key*`}` (amsmath)

тождественна `(\ref{`*key*`})`. С её помощью проще делать перекрёстные ссылки на уравнения, так как она автоматически добавляет круглые скобки вокруг ссылки, которую в LaTeX'е печатает команда `\ref` (раздел 3.7). Чтобы сослаться на уравнение с меткой `e:baset`, нужно во входном файле написать `\eqref{e:baset}`.

8.9.3. Нумерация вложенных уравнений

Бóльшую ценность представляет процедура

`\begin{subequations}` ... `\end{subequations}` (amsmath)

Если предшествующее ей уравнение имело, например, номер (4.8), то уравнения внутри неё будут иметь номера (4.9a), (4.9b), (4.9c). Разумеется, номера будут иметь только уравнения, которые бы их имели и вне процедуры `subequations`. Её тело может содержать любое количество уравнений, составленных при помощи процедур из раздела 8.6, а также любой другой текст.

Для организации такой двойной нумерации процедура `subequations` в дополнение к счётчику `equation` использует счётчик

`parentequation` (amsmath)

в котором сохраняет увеличенное на единицу значение `equation` на момент вызова процедуры. Команда

`\theparentequation` (amsmath)

печатает значение счётчика, используя определение `\theequation`, действовавшее вне процедуры. Номера уравнений внутри `subequations` и ссылки на них по-прежнему печатает команда `\theequation`, но она переопределяется следующим образом:

```
\renewcommand{\theequation}{\theparentequation\alpha{equation}}
```

Для изменения формата номеров уравнений нужно изменить `\theequation` и/или `\theparentequation` *внутри* процедуры `subequations`. В следующем примере номера уравнений (8.18a), (8.18b), (8.18c) были бы напечатаны, как (8.18а), (8.18б), (8.18в):

```
\begin{subequations}
\renewcommand{\theequation}{\theparentequation\asbuk{equation}}
...
```

Приведём пример стандартной нумерации вложенных уравнений.

$$A = B, \tag{8.18a}$$
$$D = C, \tag{8.18b}$$
$$E = F. \tag{8.18c}$$

Вся группа уравнений имеет номер (8.18), а второе уравнение имеет номер (8.18b). После завершения процедуры `subequations` восстанавливается нормальная нумерация:

$$H < K. \tag{8.19}$$

Во входном файле этот пример записан следующим образом:

```
Приведём пример стандартной нумерации вложенных уравнений.
\begin{subequations}\label{e:all}
\begin{eqnarray}
  A&=&B, \\
  D&=&C, \label{e:middle}\\
  E&=&F.
\end{eqnarray}
\end{subequations}
Вся группа уравнений имеет номер \eqref{e:all}, а второе
уравнение имеет номер \eqref{e:middle}. После завершения
процедуры \verb|subequations| восстанавливается нормальная
нумерация:
\begin{equation} H<K. \end{equation}
```

Команда `\label{e:all}`, помещённая сразу после `\begin{subequations}`, но вне других процедур, формирующих `ref`-значение, метит всю группу уравнений, так что команда `\ref{e:all}` с тем же значением ключа `e:all` печатает 14.18, а не 14.18a.

8.10. Команда \boldsymbol

Команды

`\boldsymbol{`*math*`}` `\pmb{`*math*`}`	(amsbsy, amsmath)

печатают часть формулы *math*, используя `bold` версию математических шрифтов. Обе команды определены в пакете `amsbsy`, который автоматически загружается пакетом `amsmath`. Напомним, что версию изменяет `\mathversion{`*version-name*`}` (раздел 6.6) и что имеются две версии: `normal` (используется по умолчанию) и `bold`. Напомним также, что LaTeX разрешает менять версию математических шрифтов только вне математической моды. Команда `\boldsymbol` позволяет обойти это ограничение и переключать версию шрифтов прямо внутри математической формулы. Она может печатать полужирным математическим курсивом не только буквы, но также строчные греческие буквы, на которые не действует математический алфавит `\mathbf`.

```
Сравните $\beta M$, $\mathbf{\beta M}$
и $\boldsymbol{\beta M}$.
```

Сравните βM, $\beta\mathbf{M}$ и $\boldsymbol{\beta M}$.

В современной научной литературе буквы латинского алфавита в математических формулах принято набирать прямым полужирным шрифтом (`\mathbf`), а не полужирным курсивом, который даёт команда `\boldsymbol`. Её следует использовать для строчных греческих букв или иных символов. В редких случаях, когда команда `\boldsymbol` не срабатывает (например, по причине отсутствия соответствующих шрифтов размера менее 10 pt), можно использовать команду `\pmb`, название которой происходит от слов «poor man's bold» (полужирный по бедности). Эта команда печатает с небольшими смещениями несколько копий одного и того же символа, в результате чего он становится полужирным.

8.11. «Кирпичики» формул

8.11.1. Матрицы

Пакет amsmath вводит специализированные процедуры для набора матриц в дополнение к процедуре `array`, имеющейся в формате LaTeX:

`\begin{pmatrix}`	...	`\end{pmatrix}`	
`\begin{bmatrix}`	...	`\end{bmatrix}`	
`\begin{Bmatrix}`	...	`\end{Bmatrix}`	(amsmath)
`\begin{vmatrix}`	...	`\end{vmatrix}`	
`\begin{Vmatrix}`	...	`\end{Vmatrix}`	

Они печатают матрицы соответственно в круглых скобках ( ), квадратных скобках [ ], фигурных скобках { }, в вертикальных разделителях | | и ‖ ‖. Для полноты картины добавлена процедура

```
\begin{matrix} ... \end{matrix}
```
(amsmath)

формирующая матрицу без разделителей. В отличие от `array`, все эти процедуры не имеют аргументов, указывающих количество столбцов и способ их позиционирования. Предполагается, что ячейки столбцов центрируются. Матрица может иметь до 10 столбцов или даже больше, если увеличить значение счётчика

```
MaxMatrixCols
```
(amsmath)

по умолчанию равное 10.

```
\[ \begin{Vmatrix}
  C_1+C &    -C \\
     -C & C_2+C
\end{Vmatrix} \]
```

$$\begin{Vmatrix} C_1+C & -C \\ -C & C_2+C \end{Vmatrix}$$

Если какие-то столбцы требуется выровнять по правому или левому краю, следует обратиться к процедуре `array`, упоминавшейся в главе 6. Детально она описана в разделе 12.3.

Для размещения маленькой матрицы в тексте имеется процедура

```
\begin{smallmatrix} ... \end{smallmatrix}
```
(amsmath)

Например, $\bigl(\begin{smallmatrix} a&b\\ c&d \end{smallmatrix} \bigr)$ лучше выглядит в строке, чем обычная матрица $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Разделители вокруг маленькой матрицы нужно вставлять явно, так как нет `p`-, `b`-, `v`- или `V`- версий процедуры `smallmatrix`. Встретившаяся выше маленькая матрица набрана так:

```
\begin{math}
\bigl( \begin{smallmatrix} a&b\\ c&d \end{smallmatrix} \bigr)
\end{math}
```

Команда

`\hdotsfor[`*dist*`]{`*num*`}` (amsmath)

вставляет в матрицу строку точек, занимающую указанное число столбцов *num*:

```
\begin{equation*}\begin{bmatrix}
   a & b & c & d\\
   e &\hdotsfor{3}
\end{bmatrix}    \end{equation*}
```

$$\begin{bmatrix} a & b & c & d \\ e & \hdotsfor{3} \end{bmatrix}$$

Расстояние между точками можно изменить, используя опцию *dist*. Например, `\hdotsfor[2]{4}` увеличивает его в 2 раза.

```
\begin{equation*}
\begin{pmatrix}
 D_1t&-a_{12}t_2&\cdots&-a_{1n}t_n\\
 -a_{21}t_1&D_2t&\cdots&-a_{2n}t_n\\
 \hdotsfor[2]{4}\\
 -a_{n1}t_1&-a_{n2}t_2&\dots&D_nt
\end{pmatrix}
\end{equation*}
```

$$\begin{pmatrix} D_1t & -a_{12}t_2 & \cdots & -a_{1n}t_n \\ -a_{21}t_1 & D_2t & \cdots & -a_{2n}t_n \\ \hdotsfor[2]{4} \\ -a_{n1}t_1 & -a_{n2}t_2 & \dots & D_nt \end{pmatrix}$$

8.11.2. Блоки выравнивания

Процедуры

`\begin{gathered}`[pos]	...	`\end{gathered}`	(amsmath)
`\begin{aligned}`[pos]	...	`\end{aligned}`	

создают замкнутые объекты типа матриц, которые можно использовать внутри математических выражений. Необязательный аргумент pos имеет то же назначение, что и в процедуре `array`: он позволяет изменять вертикальное позиционирование объекта по отношению к другим частям формулы. По умолчанию он имеет значение `c`, соответствующее совмещению по вертикали середины объекта с осевой линией формулы, которая проходит через перекрестие знака «+». Например:

$$\text{Позиционирование по центру} \begin{aligned} \alpha&=\alpha\alpha\\ \beta&=\beta\beta\beta\\ \gamma&=\gamma \end{aligned} , \qquad\text{по верху}\qquad \begin{aligned}[t] \delta&=\delta\delta\\ \eta&=\eta\eta\eta\\ \varphi&=\varphi \end{aligned} .$$

```
\begin{equation*}
  \text{Позиционирование по центру}
  \begin{aligned}
    \alpha&=\alpha\alpha\\ \beta&=\beta\beta\beta\\ \gamma&=\gamma
  \end{aligned} ,
  \qquad\text{по верху}\qquad
  \begin{aligned}[t]
    \delta&=\delta\delta\\ \eta&=\eta\eta\eta\\ \varphi&=\varphi
  \end{aligned} .
\end{equation*}
```

8.11.3. Условные конструкции

Условные конструкции в математике встречаются часто:

$$P_{r-j} = \begin{cases} 0, & \text{если } r-j \text{ нечётно,} \\ r!\,(-1)^{(r-j)/2}, & \text{если } r-j \text{ чётно.} \end{cases} \tag{8.20}$$

Для них вводится специальная процедура:

`\begin{cases} ... \end{cases}` (amsmath)

Она значительно упрощает запись предыдущего примера:

```
\begin{equation}
P_{r-j}=\begin{cases}
        0,                 & \text{если $r-j$ нечётно},\\
        r!\,(-1)^{(r-j)/2}, & \text{если $r-j$ чётно}.
      \end{cases}
\end{equation}
```

Здесь использованы 2 команды `\text` для печати текстовых пояснений внутри уравнения. Каждая из них сама содержит математическое выражение `$r-j$`.

8.11.4. Пробелы в формулах

Пакет amsmath несколько расширяет набор команд для установки пробелов в математических выражениях. Некоторые команды имеют краткую форму имени, состоящую из двух символов:

`\,`	`\thinspace`	┘┌	`\!`	`\negthinspace`	┘┌
`\:`	`\medspace`	┘┌		`\negmedspace`	┘┌
`\;`	`\thickspace`	┘┌		`\negthickspace`	┘┌
	`\quad`	┘ ┌			
	`\qquad`	┘ ┌			

(amsmath)

Все эти команды можно использовать также вне математических выражений. Для более точной регулировки пробелов $\mathcal{AMS}$-LaTeX имеет команду

`\mspace{`*len*`}` (amsmath)

Она является аналогом команды `\hspace{`*len*`}`. Однако в её аргументе длину можно задавать только в специальных математических единицах `mu` ($1\,\mathrm{mu} = 1/18\,\mathrm{em}$). Отрицательный пробел, равный `\quad`, можно записать как `\mspace{-18mu}`, а `\thinspace` вставляет пробел длиной $3\,\mathrm{mu}$.

8.11.5. Многоточия

Когда загружен пакет amsmath, многоточия можно набирать командой

`\dots` (amsmath)

Она совмещает свойства стандартных команд LaTeX'а `\ldots` и `\cdots`, автоматически выбирая позицию многоточия в зависимости от того, что следует за `\dots`. Если за командой `\dots` следует знак бинарной операции (например, $+$), она действует, как `\cdots`, размещая многоточие на осевой линии формулы (на уровне

дробной черты). Если следующим знаком является запятая или другой символ, многоточие будет расположено на уровне десятичной точки (как при использовании \ldots). Команда \dots особо выделяет ситуацию, когда за ней следует точка умножения или знак интеграла, которые формально относятся соответственно к категории знаков бинарных операций и других символов. В редких случаях, когда за многоточием в формуле ничего нет (следует признак конца математической моды \end, \) или $, который не несет информации относительно размещения), необходимо явно указать способ позиционирования многоточия, используя команды

```
\dotsc   \dotsb   \dotsi   \dotsm   \dotso
```
(amsmath)

соответственно для многоточия после запятой, бинарного оператора, знака интеграла, точки умножения или чего-то иного. Следующий текст во входном файле

```
Тогда мы имеем ряд $A_1, A_2 \dots A_n \dotsc$
сумму ряда $A_1+A_2+\dots+A_n+\dotsb$
произведение  $A_1\cdot A_2\dots\cdot A_n\dotsm$
и бесконечный интеграл
\[ \int_{A_1}\int_{A_2}\dotsi\int_{A_n}\dotsi \]
```

иллюстрирует практически все возможные варианты:

Тогда мы имеем ряд $A_1, A_2 \dots A_n \dotsc$ сумму ряда $A_1+A_2+\dots+A_n+\dotsb$ произведение $A_1\cdot A_2\dots\cdot A_n\dotsm$ и бесконечный интеграл

$$\int_{A_1}\int_{A_2}\dotsi\int_{A_n}\dotsi$$

8.11.6. Неразрывные дефисы

Команда

```
\nobreakdash
```
(amsmath)

обеспечивает подавление возможного переноса на следующую строку части слова после следующего за командой дефиса или тире. Например, если записать «страницы 1–9» как «страницы 1\nobreakdash--9», строка ни при каких условиях не будет разорвана между дефисом и цифрой 9. Можно также использовать \nobreakdash, чтобы предотвратить нежелательные переносы в комбинациях типа x-координата. Чтобы запретить разрыв строки после дефиса, разрешив нормальную расстановку переносов в следующем слове, достаточно добавить пробел нулевой ширины после дефиса. Например:

```
This is $3$\nobreakdash-\hspace{0pt}dimensional
problem.
```

This is 3-dimensional problem.

8.11.7. Двойные диакритические знаки

$\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX изменяет команды

$\hat{x}$	`\hat{x}`	$\acute{x}$	`\acute{x}`	$\bar{x}$	`\bar{x}`
$\check{x}$	`\check{x}`	$\grave{x}$	`\grave{x}`	$\vec{x}$	`\vec{x}`
$\breve{x}$	`\breve{x}`	$\tilde{x}$	`\tilde{x}`		
$\dot{x}$	`\dot{x}`	$\ddot{x}$	`\ddot{x}`		

(amsmath)

для расстановки диакритических знаков над символами. Эти команды имеют те же имена, что и в формате LaTeX, но они обеспечивают более качественное позиционирование двойных диакритических знаков. Сравните два изображения буквы A с двойной шляпкой (`\hat{\hat{A}}`) в размере `\Huge`:

$$\hat{\hat{A}} \text{ и } \hat{\hat{A}}$$

Левое получено без загрузки пакета amsmath, правое — с этим пакетом.

Пакет amsxtra имеет декларацию[3]

`\accentedsymbol{`*cmd*`}{`*def*`}` (amsxtra)

для определения символов с диакритическими знаками. Она работает аналогично декларации `\newcommand` (сохраняя смысл аргументов *cmd* и *def*), но подготовленные с её помощью символы печатаются мгновенно, так как LaTeX запоминает их изображение, а не способ изготовления, как это было бы в случае с `\newcommand`. Вот как следует изготовить команду `\Ahathat` для символа $\hat{\hat{A}}$:

```
\accentedsymbol{\Ahathat}{\hat{\hat A}}
```

Команды

$\dddot{x}$	`\dddot{x}`	$\ddddot{x}$	`\ddddot{x}`

(amsmath)

производят диакритические знаки в виде трёх и четырёх точек в дополнение к командам `\dot` и `\ddot`, доступным в формате LaTeX.

8.11.8. Корни

Команды

`\leftroot{`*num*`}  \uproot{`*num*`}` (amsmath)

[3] Помимо `\accentedsymbol` пакет amsxtra содержит определения ещё нескольких команд: `\fracwithdelims`, `\sphat`, `\spcheck`, `\sptilde`, `\spdot`, `\spddot`, `\spdddot`, `\spbreve`. Мы их не описываем, так как они считаются устаревшими.

позволяют скорректировать размещение показателя корня, если оно неудовлетворительно, сдвигая показатель соответственно влево и вверх при положительном значении аргумента *num* или вправо и вниз при отрицательном. Абсолютная величина параметра *num* задаёт количество шагов сдвига, а величина шага (достаточно маленькая) зависит от размера шрифта.

```
Сравните $\sqrt[\beta]{k}$ и
$\sqrt[\leftroot{-2}\uproot{2}\beta]{k}$.
```

Сравните $\sqrt[\beta]{k}$ и $\sqrt[\leftroot{-2}\uproot{2}\beta]{k}$.

8.11.9. Формулы в рамке

Команда

`\boxed{`*math*`}` (amsmath)

рисует рамку вокруг формулы *math*, стоящей в её аргументе.

```
\begin{equation}    \boxed{
  \eta \leq C(\delta(\eta)
  +\Lambda_M(0,\delta))   }
\end{equation}
```

$$\boxed{\eta \leq C(\delta(\eta)+\Lambda_M(0,\delta))} \qquad (8.21)$$

В отличие от аналогичной команды `\fbox`, имеющейся в формате LaTeX (раздел 9.1), её аргумент *math* обрабатывается в математической моде.

8.11.10. Размещение объектов друг над другом

LaTeX имеет команды `\overrightarrow` и `\overleftarrow`, которые рисуют стрелки над тем, что стоит в их аргументе. $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX вводит ещё несколько стрелок, так что полный набор содержит 6 команд:

`\overleftarrow{`*math*`}`	`\underleftarrow{`*math*`}`
`\overrightarrow{`*math*`}`	`\underrightarrow{`*math*`}`
`\overleftrightarrow{`*math*`}`	`\underleftrightarrow{`*math*`}`

(amsmath)

Назначение каждой команды легко расшифровать, зная, что `left` — это налево, `right` — направо, `under` — снизу, `over` — сверху.

```
$\overleftrightarrow{fgh+
 \underleftarrow{abc\mathstrut}}$
```

$\overleftrightarrow{fgh+\underleftarrow{abc\mathstrut}}$

Здесь для регулировки положения стрелки по высоте команда `\mathstrut` вставляет страту, то есть невидимый символ, имеющий высоту круглой скобки «(» и нулевую ширину (раздел 6.9).

Команды

`\xleftarrow[`*subscript*`]{`*superscript*`}`
`\xrightarrow[`*subscript*`]{`*superscript*`}` (amsmath)

производят стрелки, которые растягиваются автоматически так, чтобы над ними или под ними могли разместиться необычно широкие индексы. У этих команд один обязательный аргумент *superscript* (верхний индекс) и один необязательный аргумент *subscript* (нижний индекс).

```
\begin{equation*}
  A\xleftarrow{n+\mu-1}B
  \xrightarrow[T]{n\pm i-1}C
\end{equation*}
```

$$A \xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1} C$$

Формат LaTeX имеет команду `\stackrel` для размещения верхнего индекса над бинарным оператором. Пакет amsmath вводит ещё две команды аналогичного действия

`\underset{`*subscript*`}{`*math*`}`
`\overset{`*superscript*`}{`*math*`}` (amsmath)

Они помещают индекс, находящийся в первом аргументе, соответственно под и над формулой или символом, стоящим в аргументе *math*. Например, `$\underset {*}{X}$` печатает $\underset{*}{X}$, а `$\overset{*}{X}$` — $\overset{*}{X}$ (как и `$\stackrel{*}{X}$`). Методы форматирования многострочных индексов мы рассмотрим в разделе 8.11.17.

8.11.11. Дроби и биномы

В дополнение к команде LaTeX'а `\frac` для построения дробей AMS-LaTeX вводит команды

`\dfrac{`*numerator*`}{`*denumerator*`}`
`\tfrac{`*numerator*`}{`*denumerator*`}` (amsmath)

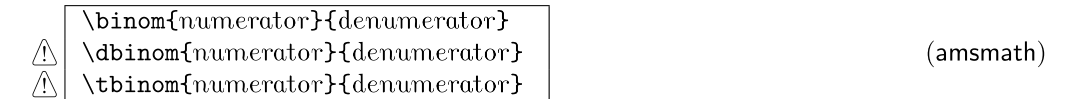
как аббревиатуры для `{\displaystyle\frac...}` и `{\textstyle\frac...}`. Иными словами, команда `\frac` эквивалентна `\dfrac` в выключной формуле и `\tfrac` в обычной формуле, размещаемой среди текста.

```
\begin{equation}
  \frac{1}{k}\log_2 c(f),  \quad
  \tfrac{1}{k}\log_2 c(f).
\end{equation}
$\sqrt{\frac{1}{k}\log_2 c(f)} $,
$\sqrt{\dfrac{1}{k}\log_2 c(f)}$.
```

$$\frac{1}{k}\log_2 c(f), \quad \tfrac{1}{k}\log_2 c(f). \tag{8.22}$$

$\sqrt{\frac{1}{k}\log_2 c(f)}$, $\sqrt{\dfrac{1}{k}\log_2 c(f)}$.

Для набора биномиальных выражений типа $\binom{n}{k}$ применяются команды



`\binom{`*numerator*`}{`*denumerator*`}`
⚠ `\dbinom{`*numerator*`}{`*denumerator*`}` (amsmath)
⚠ `\tbinom{`*numerator*`}{`*denumerator*`}`

Их аргументы {*numerator*} и {*denumerator*} имеют смысл числителя и знаменателя, как и у команды `\frac`.

```
\begin{equation}
  2^k-\binom{k}{1}2^{k-1}+
  \dbinom{k}{2}2^{k-2}+
  \tbinom{k}{2}2^{k-3}
\end{equation}
```

$$2^k - \binom{k}{1}2^{k-1} + \dbinom{k}{2}2^{k-2} + \tbinom{k}{2}2^{k-3} \qquad (8.23)$$

Команды `\frac`, `\binom` являются частным случаем `\genfrac` с шестью аргументами:

`\genfrac{`*left-delim*`}{`*right-delim*`}{`*thickness*`}{`*mathstyle*`}{`*numerator*`}{`*denumerator*`}` (amsmath)

Первые два аргумента определяют вид разделителей вокруг дроби (как, например, в `\binom`). Они должны быть пусты, когда разделители не нужны. Третий аргумент *thickness* определяет толщину дробной черты (в `\binom` она равна нулю). Четвертый аргумент *mathstyle* должен быть целым числом в диапазоне от 0 до 3 в зависимости от выбранного стиля форматирования дроби: `\displaystyle`, `\textstyle`, `\scriptstyle` или `\scriptscriptstyle`; если он пуст, стиль дроби определяется из контекста. Последние два аргумента есть собственно числитель и знаменатель дроби. Если третий аргумент *thickness* оставлен пустым, толщина дробной черты устанавливается по умолчанию. Покажем, как могли бы быть определены `\frac`, `\tfrac` и `\binom` в терминах команды `\genfrac`:

```
\newcommand{\frac}[2]{\genfrac{}{}{}{}{#1}{#2}}
\newcommand{\tfrac}[2]{\genfrac{}{}{}{1}{#1}{#2}}
\newcommand{\binom}[2]{\genfrac{(}{)}{0pt}{}{#1}{#2}}
```

Ещё одна команда

`\cfrac[`*pos*`]{`*numerator*`}{`*denumerator*`}` (amsmath)

предназначена для набора непрерывных дробей.

```
\begin{equation}
\cfrac{1}{\sqrt{2}+
 \cfrac{1}{\sqrt{2}+
  \cfrac{1}{\sqrt{2}+\cdots}}}
\end{equation}
```

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{2}+\cdots}}} \qquad (8.24)$$

Выравнивание числителя любой из дробей влево или вправо достигается командой `\cfrac` с опцией *pos*. Она может принимать значения `l` или `r` (ср. раздел 6.5).

8.11.12. Опции команды \smash

Команда

`\smash[`*pos*`]{`*math*`}` (amsmath)

используется для «обмана». LaTeX форматирует формулу *math* в аргументе команды так, как если вместо `\smash[pos]{math}` просто стоял бы её обязательный аргумент `{math}` (вместе с фигурными скобками), но позиционирует её так, будто она имеет нулевую высоту и/или глубину. Высота и глубина отсчитываются от базисной линии формулы, где располагается десятичная точка. Иногда этот «обман» полезен для корректировки взаимного расположения частей сложной математической формулы. Команда `\smash` имеется и в формате LaTeX (раздел 6.9), но пакет amsmath расширяет её возможности, добавляя необязательный аргумент *pos*, который может иметь значение `t` или `b`. В первом случае (*pos*=`t`) обнуляется высота формулы, во втором случае (*pos*=`b`) — её глубина при сохранении естественной высоты. Например, когда подкоренные выражения имеют неодинаковые размеры, использование команды `\smash` сделает структуру формулы более однородной.

```
Сравните $\sqrt{x}+\sqrt{y}+\sqrt{z}$ и
$\sqrt{x}+\sqrt{\smash[b]{y}}+\sqrt{z}$.
```

Сравните $\sqrt{x}+\sqrt{y}+\sqrt{z}$ и $\sqrt{x}+\sqrt{\smash[b]{y}}+\sqrt{z}$.

8.11.13. Разделители

Вертикальная черта |, принадлежащая по классификации LaTeX'а к группе разделителей, в современной математике используется для обозначения самых разнообразных объектов. В теории чисел используют выражения вида $p|q$. Более известны обозначение для абсолютного значения $|z|$ и запись вида $f_\zeta(t)\big|_{t=0}$, означающая, что значение функции $f_\zeta(t)$ требуется вычислить при $t = 0$. Применение одного и того же символа для разных целей само по себе не так уж плохо. Плохо то, что не все эти применения имеют адекватное обозначение во входном файле. По крайней мере, следует различать правые и левые формы разделителей | и ‖. AMS-LaTeX вводит для этой цели две пары команд:

$\lvert$	`\lvert`	$\rvert$	`\rvert`
$\lVert$	`\lVert`	$\rVert$	`\rVert`

(amsmath)

Теперь легко определить команды для набора абсолютного значения и нормы (длины) вектора:

```
\newcommand{\abs}[1]{\lvert#1\rvert}
\newcommand{\norm}[1]{\lVert#1\rVert}
```

В результате `$\abs{z}\cdot\norm{A}$` напечатает $|z| \cdot \|A\|$.

8.11.14. Имена операторов

Математические функции типа log, sin и lim традиционно печатаются прямым шрифтом, чтобы отличить их от математических переменных, которые воспроизводятся математическим курсивом. Имена наиболее известных функций предопределены в формате LaTeX, но в математических рукописях всё время появляются новые обозначения функций. Пакет amsopn предлагает общий механизм

для определения нового имени «оператора». Поскольку amsopn автоматически загружается пакетом amsmath, последнему также доступен этот механизм. Чтобы определить обозначение функции, следует использовать декларацию (она должна находиться в преамбуле)

```
\DeclareMathOperator{cmd}{def}
\DeclareMathOperator*{cmd}{def}
```
(amsopn)

где cmd — имя новой команды, а def — текст, который будет напечатан в качестве имени функции (добавлять декларации, форматирующие имя, не нужно!).

```
\DeclareMathOperator{\xxx}{xxx}
  ...
\[ A\xxx(B)\neq A\mathrm{xxx}(B) \]
```

...

$$A\operatorname{xxx}(B) \neq A\mathrm{xxx}(B)$$

Вновь определённая команда `\xxx` в этом примере расставляет правильные пробелы вокруг `xxx`. Во втором аргументе def преобладает псевдотекстовый режим: дефис «-» будет печататься как текстовый дефис, но не как знак «минус», а звёздочка «*» будет приподнята (в математической моде звёздочка центрируется). С другой стороны, имя вновь введённой функции обрабатывается в математической моде, поэтому с ним можно использовать нижние и верхние индексы.

Если у нового оператора, когда он находится в выключной формуле, индексы должны размещаться в позиции предела сверху или снизу, как у $\lim\limits_{t\to0}$, следует использовать *-форму декларации `\DeclareMathOperator`:

```
\DeclareMathOperator*{\Xxx}{xxx}
  ...
\[ \Xxx_{x\to0}\neq \xxx_{x\to0}  \]
```

...

$$\operatorname*{xxx}_{x\to0} \neq \operatorname{xxx}_{x\to0}$$

Обычно размещение индексов можно изменить при помощи `\nolimits` и `\limits`. $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX значительно ограничивает сферу действия этих команд. Например, ни при каких условиях он не позволит для функции sin (произведённой командой `\sin`) запись вида $\sin\limits_{t\to0}$, поскольку та не имеет смысла в отличие от $\lim\limits_{t\to0}$.

```
...запись вида $\sin\limits_{t\to0}$
не имеет смысла в отличие от
$\lim\limits_{t\to0}$.
```

...запись вида $\sin_{t\to0}$ не имеет смысла в отличие от $\lim\limits_{t\to0}$.

Команды, определённые посредством `\DeclareMathOperator`, относятся к классу функций `\sin`; положение их индексов невозможно изменить при помощи `\limits` или `\nolimits`. Напротив, *-форма декларации `\DeclareMathOperator*` вводит команды типа `\lim`, которые не игнорируют указания `\limits` и `\nolimits`. Всё сказанное относится к случаю, когда пакет amsopn или amsmath загружен с опцией `namelimits` (действует по умолчанию). Опция `nonamelimits` приравнивает все функции к классу `\sin`.

Есть также команды

```
\operatorname{def}
\operatorname*{def}
```
(amsopn)

такие, что употребление `\operatorname{abc}` в математической формуле эквивалентно команде `\abc`, ранее определённой посредством `\DeclareMathOperator {\abc}{abc}`.

Небольшое количество специальных имен функций предопределены в пакете amsopn в дополнение к имевшимся в формате LaTeX:

$\varinjlim$	`\varinjlim`	$\varprojlim$	`\varprojlim`
$\varliminf$	`\varliminf`	$\varlimsup$	`\varlimsup`
$\projlim$	`\projlim`	$\injlim$	`\injlim`

(amsopn)

8.11.15. Команда `\mod` и её производные

Команды



`\mod{`*math*`}`	`\bmod`
`\pod{`*math*`}`	`\pmod{`*math*`}`

(amsmath)

предназначены для обозначения операции выделения остатка по модулю. Команды `\bmod` и `\pmod` доступны в LaTeX'е, но пакет amsopn несколько изменяет пробелы вокруг mod, когда эти команды используются в формуле внутри текста. Команды `\mod` и `\pod` суть варианты `\pmod`, предпочитаемые некоторыми авторами; `\mod` опускает круглые скобки, в то время как `\pod` опускает mod, но сохраняет круглые скобки.

```
$\gcd(n,m\bmod n)$;\\
$x\equiv y\pmod b$;\\
$x\equiv y\mod c$;\\
$x\equiv y\pod d$.
```

$\gcd(n,m\bmod n)$;
$x\equiv y\pmod b$;
$x\equiv y\mod c$;
$x\equiv y\pod d$.

8.11.16. Знаки кратных интегралов

Команды

$\iint$	`\iint`	$\iiiint$	`\iiiint`
$\iiint$	`\iiint`	$\idotsint$	`\idotsint`

(amsmath)

печатают знаки кратных интегралов с правильно подобранными интервалами между символами $\int$.

```
Сравните $\int\int\int\limits_A$ и $\iiint\limits_A$.
```

Сравните $\int\int\int\limits_A$ и $\iiint\limits_A$.

8.11.17. Многострочные индексы

Команда

`\substack{`*lines*`}`

(amsmath)

предназначена для набора многострочных индексов. Обычно индексы в несколько строчек приходится использовать в качестве пределов суммирования. Строки в аргументе *lines* разделяются командами \\:

```
\sum_{\substack{i\in\Lambda
      \\ 0<j<n}} P(i,j)
```

$$\sum_{\substack{i\in\Lambda \\ 0<j<n}} P(i,j)$$

Если строчки в индексах необходимо выровнять по левому или правому краю, проще всего использовать процедуру

`\begin{subarray}{`*pos*`} ... \end{subarray}` (amsmath)

Её аргумент *pos* может принимать значения `l`, `c` или `r` соответственно тому, нужно ли выравнивать строки относительно левого края, центра или правого края. В следующем примере строки выровнены по левому краю:

```
\sum_{\begin{subarray}{l}
        i\in\Lambda\\ 0<j<n
      \end{subarray}} P(i,j)
```

$$\sum_{\begin{subarray}{l} i\in\Lambda\\ 0<j<n \end{subarray}} P(i,j)$$

У `\substack` и `subarray` есть аналог — команда `\shortstack`, действующая в текстовой моде (раздел 9.6.4), причём она совмещает простоту команды `\substack` со способностью процедуры `subarray` выравнивать строки.

8.11.18. Сторонние индексы

Команда

`\sideset{`*left-scripts*`}{`*right-scripts*`}` (amsmath)

размещает индексы по углам символа переменного размера типа $\sum$ или $\prod$.

```
Сравните
\[
\sideset{_1^2}{_4^3}\prod
\text{ и } {_1^2\prod_4^3}.
\]
```

Сравните

$$\sideset{_1^2}{_4^3}\prod \text{ и } {_1^2\prod_4^3}.$$

Рассмотрим более содержательный пример. Допустим, что необходимо пометить штрихом знак суммы. Если у знака суммы нет пределов суммирования, достаточно добавить `\nolimits` перед штрихом:

```
\[ \sum\nolimits' E_n \]
```

$$\sum\nolimits' E_n$$

Однако при наличии пределов суммирования эта технология не приводит к удовлетворительному результату:

```
\[ \sum_{n<k}\nolimits' E_n \]
```

$$\sum_{n<k}\nolimits' E_n$$

Решает проблему команда `\sideset`:

```
\[ \sideset{}{'}\sum_{n<k} E_n \]
```

$$\sideset{}{'}\sum_{n<k} E_n$$

Дополнительная пара пустых фигурных скобок объясняется тем, что индексов слева от знака суммы в данном случае не должно быть.

8.12. Коммутативные диаграммы

Коммутативными диаграммами называются схемы вида

$$\begin{CD} S^{{\mathcal{W}}_\Lambda}\otimes T @>j>> T \\ @VVV @VV{\lim P}V \\ (S\otimes T)/I @= (Z\otimes T)/J \end{CD}$$

Простейшие коммутативные диаграммы (без диагональных стрелок) формирует процедура

`\begin{CD} ... \end{CD}` (amscd)

определённая в пакете amscd из коллекции $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX. Коммутативная диаграмма, приведённая выше, набрана следующим образом:

```
\begin{equation*}\begin{CD}
  S^{{\mathcal{W}}_\Lambda}\otimes T   @>j>>  T             \\
  @VVV                                        @VV{\lim P}V  \\
  (S\otimes T)/I                       @=     (Z\otimes T)/J
\end{CD}\end{equation*}
```

В процедуре CD обозначения

`@>> @<< @VVV @AAA @=` (amscd)

производят стрелки соответственно вправо, влево, вниз и вверх, а также двойную горизонтальную линию. В случае горизонтальных стрелок `@>>>` или `@<<<` материал между первым и вторым символом `>` или `<` будет напечатан в верхнем индексе, а материал между вторым и третьим символом будет помещен в нижнем индексе. Аналогично материал между первым и вторым (вторым и третьим) символами `V` или `A` в обозначении вертикальных стрелок `@VVV` или `@AAA` будет напечатан слева (справа) от стрелки.

8.13. Теоремы и теоремоподобные структуры

Пакет amsthm расширил возможности декларации LaTeX'а `\newtheorem` (раздел 7.3) по конструированию теоремоподобных процедур, добавив `*`-форму этой

декларации для определения теорем, которые не нужно нумеровать. Расширенная версия `\newtheorem` варьирует стиль форматирования теорем в соответствии с указанным декларацией `\theoremstyle`. Пакет amsthm вводит также процедуру `proof` (доказательство), которая автоматически добавляет символ □ в конце доказательства.

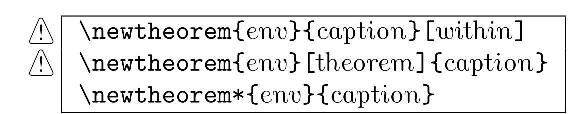
8.13.1. Теоремы

Математические трактаты, как правило, формулируют теоремы и содержат их доказательства. Часто также даются формулировки лемм, аксиом, определений, предложений, суждений, замечаний, случаев и т. д. Поскольку все такие теоремоподобные структуры формируют из текстового потока абзацы с хорошо очерченными границами, их естественно оформлять в виде процедур LaTeX'а. Стандартные классы печатных документов LaTeX'а не могут предусмотреть все зигзаги пытливого ума ученых математиков. Однако взамен они предоставляют автору текста средство для конструирования отсутствующих процедур в виде декларации `\newtheorem`:

```
⚠ \newtheorem{env}{caption}[within]
⚠ \newtheorem{env}[theorem]{caption}
  \newtheorem*{env}{caption}
```
(amsthm)

Мы не будем напоминать смысл аргументов *env*, *caption*, *theorem* и *within*, отсылая Читателя за подробностями и примерами к разделу 7.3, так как первые два из перечисленных трёх вариантов `\newtheorem` имеются в формате LaTeX'а. Пакет amsthm добавляет `*`-форму декларации `\newtheorem`, которая вводит процедуру *env*, не использующую автоматическую нумерацию. В результате её применения пример со стр. 165 теперь выглядит так:

```
\newtheorem*{Fermat}{Теорема Ферма}
\begin{Fermat}
Нет целых чисел $n>2$, $x$, $y$
и $z$ таких, что $x^n+y^n=z^n$.
\end{Fermat}
```


Теорема Ферма. *Нет целых чисел* $n > 2$, x, y *и* z *таких, что* $x^n + y^n = z^n$.

8.13.2. Стиль теоремы

Пакет `amsthm` вводит понятие «стиль теоремы» и соответствующую декларацию

```
\theoremstyle{style}
```
(amsthm)

которая осуществляет выбор стиля *style*. От этого выбора зависит, как будет оформлен текст теоремы. Существуют три стиля: `plain`, `definition` и `remark`. Стиль `plain`, используемый по умолчанию, печатает текст курсивом, тогда как `definition` и `remark` курсив не используют. Другие детали оформления теоремы могут изменяться в зависимости от выбора класса печатного документа.

Теоремоподобная процедура использует тот стиль, который действовал на момент её определения. Следовательно, декларация `\theoremstyle` должна предшествовать `\newtheorem`. На практике следует собрать декларации `\newtheorem` в одном месте входного файла (лучше всего в преамбуле), разделить на группы и перед каждой группой вставить `\theoremstyle{`*style*`}`.

```
\theoremstyle{plain}
\newtheorem{Theorem}{Теорема}[section]  %
\newtheorem{Lemmma}[Theorem]{Лемма}
\newtheorem{Proposition}[Theorem]{Утверждение}
\newtheorem*{Fermat}{Теорема Ферма}

\theoremstyle{definition}
\newtheorem{Axiom}{Аксиома}[section]
\newtheorem{Sequence}{Следствие}[section]
\newtheorem{Example}{Пример}[section]

\theoremstyle{remark}
\newtheorem*{Remark}{Замечание}
\newtheorem{Case}{Случай}
```

В этом примере новые процедуры `Theorem`, `Lemmma` и `Proposition` определены так, что будут использовать стиль `plain` и будут нумероваться единым счётчиком `Theorem`. Следующая группа процедур: `Axiom`, `Sequence` и `Example` — использует стиль `definition`, а их нумерация независима в пределах каждого раздела, начинающегося с команды `\section`. Наконец, процедуры `Remark` и `Case` оформляются в стиле `remark`. Процедура `Remark` не нумеруется, а `Case` имеет сплошную нумерацию в пределах всего печатного документа.

Если `\theoremstyle` вообще отсутствует, используется стиль `plain`.

Ещё один вариант стиля теорем, разрешаемый пакетом amsthm, заключается в возможности перестановки номера теоремы и её заголовка (аргумент *caption* декларации `\newtheorem`). В результате перестановки номер теоремы печатается слева от заголовка, а не справа. Перестановка будет осуществлена, если вставить декларацию

`\swapnumbers` (amsthm)

перед списком деклараций `\newtheorem`, на которые нужно воздействовать. В результате следующих определений

```
\theoremstyle{definition}
{\swapnumbers \newtheorem{Axiom}{Аксиома}[section]}
\newtheorem{Sequence}{Следствие}[section]
```

заголовки Аксиомы и Следствия будут напечатаны в форме «**1.1. Аксиома.**» и «**Следствие 1.1.**»

8.13.3. Доказательства

Процедура

`\begin{proof}[caption] ... \end{proof}` (amsthm)

форматирует доказательства теорем.

```
\begin{proof}
...следовательно $G(t)=L\gamma!\,
t^{-\gamma} + t^{-\delta}\eta(t)$
\end{proof}
```

Доказательство. ...следовательно $G(t) = L\gamma!\,t^{-\gamma} + t^{-\delta}\eta(t)$ □

Она печатает заголовок «Доказательство» (или «Proof», если опция `russian` не указана при загрузке пакета babel), а в конце доказательства ставит знак □ («что и требовалось доказать»). Процедура `proof` предназначается для коротких доказательств, занимающих не более одной-двух страниц. Более длинные доказательства целесообразно оформить в виде специального раздела, используя команды секционирования `\section` или `\subsection`.

Заголовок процедуры `proof`, печатаемый по умолчанию, хранится в команде

`\proofname` (amsthm)

При необходимости `\proofname` можно изменить при помощи `\renewcommand`. Например:

```
\renewcommand{\proofname}{Доказательство}
```

Необязательный аргумент *caption* процедуры `proof` служит примерно той же цели. Он позволяет заменить текущее значение заголовка на какое-нибудь иное, например:

```
\begin{proof}[Доказательство Основной Теоремы]
```

Однако на следующий вызов процедуры `proof` это значение заголовка, естественно, не распространяется.

Признак конца доказательства □ хранится в команде

`\qedsymbol` (amsthm)

Её также можно переопределить посредством `\renewcommand`. Для длинного доказательства, напечатанного без использования процедуры `proof`, символ □ вместе со стандартным пробелом перед ним можно набрать, используя команду

`\qed` (amsthm)

Размещение символа □ может не удовлетворить придирчивого Читателя, если доказательство заканчивается выключным уравнением или чем-то в этом роде. Выход из затруднительного положения можно найти, разместив `\qed` в подходящем месте доказательства.

```
\begin{proof}
...следовательно
\begin{equation}
 G(t)=L\gamma!\,t^{-\gamma} +
 t^{-\delta}\eta(t)           \qed
\end{equation}
\renewcommand{\qed}{}\end{proof}
```

Доказательство.
…следовательно

$$G(t) = L\gamma!\, t^{-\gamma} + t^{-\delta}\eta(t) \quad \square \qquad (8.25)$$

Здесь `\renewcommand{\qed}{}` «обнуляет» команду `\qed` перед завершением процедуры `proof`, чтобы она не напечатала признак конца доказательства повторно. Эффект «обнуления» локализован командной скобкой `\end` между границами процедуры `proof` по обычным правилам группирования, действующим в LaTeX'е (раздел 2.4).

Бокс — это искусство или спорт?
Архив КВН

Глава 9

Боксы и что там внутри

В переводе с английского языка слово *box* (бокс) означает ящик. По терминологии LaTeX'а *боксом* называется прямоугольник, который независимо от его размеров и содержимого не может быть расщеплён на части, и поэтому его нельзя по частям перенести на следующую строку или страницу. Например, LaTeX считает, что каждая буква упакована в свой маленький ящик-бокс, как показано на рис. 9.1. Буква может вылезать за пределы бокса, так что бокс и изображение, заключённое в нём,— это совсем разные вещи. Боксы — это кирпичики, из которых LaTeX строит здание печатного документа. Для успешного строительства LaTeX должен знать только размеры кирпичиков-боксов, а не то, как устроен каждый из них. Каждый бокс имеет *точку привязки* (reference point). Если бокс не вращали, точка привязки расположена на его левой грани. Через точку привязки проходит *базисная линия* бокса (baseline). Выстраивая из букв слова и строки, LaTeX размещает боксы с запакованными в них буквами так, что все точки привязки располагаются на одной базисной линии, а вертикальные грани боксов-букв в слове накрепко склеены, как показано на рис. 9.2. Аналогичным образом LaTeX поступает с боксами, содержащими любые другие объекты, например рисунки. Если между двумя боксами с рисунками нет пробела в исходном тексте, то они будут склеены так же, как две соседние буквы в слове.

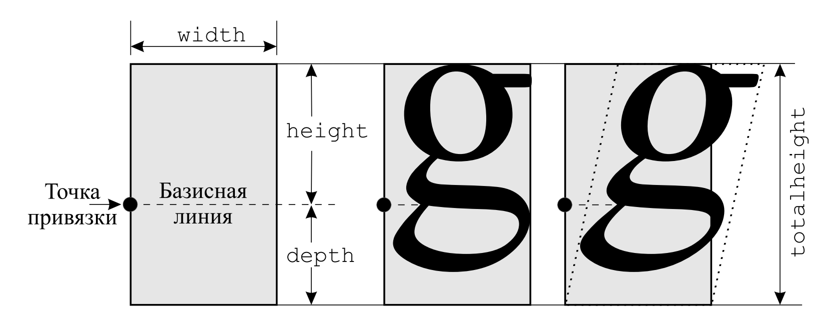


Рис. 9.1. Бокс в LaTeX'е. Изображение буквы g справа выходит за пределы бокса

Каждый бокс характеризуется *высотой*, *глубиной* и *шириной*, которые обозначаются соответственно как `height`, `depth` и `width`. Смысл этих параметров

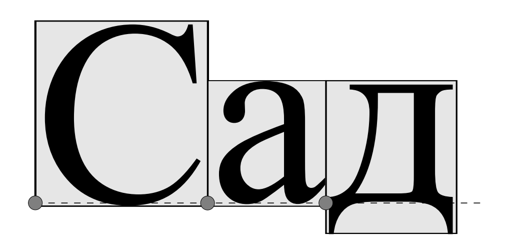


Рис. 9.2. Склейка букв в слово

должен быть понятен из рис. 9.1. Полная высота бокса `totalheight` есть расстояние между его верхней и нижней гранями, равное сумме `height` и `depth`.

В предыдущих главах мы уже встречали примеры боксов больших размеров. Такие боксы производит процедура `array` (раздел 6.4.5), используемая для набора матриц в математических формулах. Неоднократно упоминавшаяся команда `\mbox` производит так называемые строковые боксы. Всего же существуют боксы четырёх видов:

- строковые боксы (LR[1] боксы), в которых текст обрабатывается в текстовой моде, но не разбивается на строки;
- текстовые боксы (парбоксы), в которых LaTeX работает в текстовой моде;
- линейные боксы (плашки), представляющие собой чёрные прямоугольники;
- графические боксы (рисунки), образуемые процедурой `picture`, о которой мы расскажем в данной главе.

Команды и процедуры, формирующие боксы, могут использоваться в любом режиме: текстовом, строковом, математическом, графическом. Приступая к форматированию текста в боксах, LaTeX применяет декларации, действовавшие на этот момент. Исключение составляют боксы в математических формулах. Поскольку специальный математический шрифт применяется только в формулах, внутри текстового (строкового) бокса восстанавливается шрифт, использовавшийся непосредственно перед переключением LaTeX'а в математическую моду. Фрагмент текста, переданный в бокс, обычно содержится в аргументе каких-либо команд или теле процедур, поэтому декларации внутри этого фрагмента локальны для соответствующего бокса.

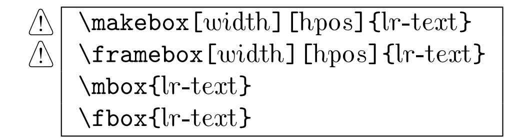
Бокс большого размера часто размещают в отдельной строке, для чего используют процедуру `center` (раздел 5.1). Ценителям оригинального жанра можно рекомендовать процедуру `displaymath`, которая допускает краткую запись `\[...\]` (раздел 6.1). Очень большие текстовые боксы и рисунки размещают как плавающие объекты посредством процедур `table` или `figure` (глава 11).

Рассмотрим теперь каждый тип боксов в отдельности.

[1] От английского Left to Right (слева направо).

9.1. Строковые боксы

Команды



```
\makebox[width][hpos]{lr-text}
\framebox[width][hpos]{lr-text}
\mbox{lr-text}
\fbox{lr-text}
```

печатают текст, обозначенный здесь как *lr-text*, в строковой моде. Команды `\framebox` и `\fbox` отличаются от двух других тем, что обводят бокс рамкой. При отсутствии опций команды `\makebox` и `\framebox` совершенно идентичны соответственно командам `\mbox` и `\fbox`. Команды `\makebox`, `\framebox` хрупкие, как и большинство команд с необязательными аргументами.

Ширина бокса подбирается равной естественной ширине текста, содержащегося в *lr-text*, за исключением случая, когда у команд `\makebox` или `\framebox` указан необязательный аргумент *width*. Нерастяжимая положительная длина *width* может быть задана в явном виде (например, `20mm`) или в долях естественных размеров текстового бокса. Естественные ширина, высота (над базисной линией), глубина (под базисной линией) и сумма высоты и глубины бокса доступны в виде команд

```
\width    \height    \depth    \totalheight
```

но только внутри аргумента *width*.

При наличии первой опции *width* позиционирование текста в боксе определяет второй необязательный аргумент *hpos*, который может состоять из одной буквы: `l`, `c`, `r` или `s`, причём

- `l` сдвигает текст к левому краю бокса,
- `c` размещает текст в центре бокса,
- `r` сдвигает текст к правому краю бокса,
- `s` растягивает текст на всю ширину бокса.

Растяжение текста при наличии опции `s` производится за счёт изменения любых растяжимых длин в аргументе *lr-text*, в том числе пробелов между словами. Если таких растяжимых длин в *lr-text* нет, то возможно появление предупреждения `Underfull hbox`[2]. По умолчанию действует опция `c`.

Перейдём к примерам.

Применение команды `\mbox` для предотвращения переноса слов по слогам мы обсуждали в разделе 4.4. Её расширенный аналог, то есть команда `\makebox`, даёт возможность заставить компилятор думать, будто слово имеет длину, которая больше естественной:

```
Это \makebox[20mm]{пример} бокса.\\
Это \makebox[20mm][r]{пример} бокса.\\
Это \makebox[20mm][l]{пример} бокса.
```

Это   пример   бокса.
Это   пример бокса.
Это пример   бокса.

[2] Незаполненный бокс.

Если заданная ширина бокса меньше естественной ширины текста, текст выйдет за границы бокса. Вот что получается, если центрировать узкие боксы:

```
{\centering
\framebox[1mm]{пример пример}\\
\framebox[1mm][r]{пример пример}\\
\framebox[1mm][l]{пример пример}}
```

пример|пример
пример пример|
|пример пример

Здесь для наглядности использована команда `\framebox` вместо `\makemebox`. Она, как и команда `\fbox`, рисует рамку вокруг бокса, причём высота и положение рамки зависят от содержания бокса:

```
\fbox{abc} \fbox{efg} \fbox{xyz}
```

abc efg xyz

В разделе 9.1.2 мы покажем, как можно регулировать высоту бокса.

Следующий пример демонстрирует способ указания ширины бокса в долях естественной ширины текста при одновременном использовании опции `s` для увеличения пробелов между словами. Сравните:

```
\framebox[1.4\width][s]{abc efg xyz} и\\
\framebox[1.4\width][c]{abc efg xyz}
```

abc efg xyz и
abc efg xyz

Чтобы обвести букву «f» квадратной рамкой, следует приравнять её ширину высоте бокса `\totalheight`. Сравните:

```
\fbox{f} и \fbox{\makebox[\totalheight]{f}}
```

f и f

Величина `\totalheight` включает в себя также толщину рамки и толщину зазора между рамкой и текстом.

Узкие боксы удобно использовать для выравнивания текста в ячейках таблиц (глава 12). Представим, например, что дробная часть числа 3,14 помещена в бокс нулевой ширины, т. е. число записано в виде `3\makebox[0mm][l]{,14}`. Бокс нулевой ширины представляет собой невидимую вертикальную линию. При использовании опции `l` (или `r`), левый (правый) край текста находится на этой линии, а текст располагается справа (слева) от неё. В результате число 3,14 будет выровнено так, будто бы его правый край располагался между цифрой 3 и запятой.

9.1.1. Измерение боксов

Декларации

```
\settowidth{cmd}{lr-text}
\settoheight{cmd}{lr-text}
\settodepth{cmd}{lr-text}
```

присваивают команде *cmd* результат измерения соответственно естественной ширины, естественной высоты и естественной глубины бокса, который получается,

если аргумент lr-text обработать в строковой моде (то есть без разбиения даже очень длинного текста на строки). Например,

```
\settowidth{\abc}{\textsc{жираф}}
```

присваивает командной длине `\abc` значение, равное ширине слова ЖИРАФ, набранного капителью. В данном случае предполагается, что команда `\abc` была ранее объявлена при помощи декларации `\newlength{\abc}` (раздел 2.10).

Следующий пример показывает, как вокруг разных слов нарисовать рамки с шириной, равной ширине самого длинного слова:

```
\newlength{\abc}\settowidth{\abc}{жираф}
\framebox[\abc]{слон} \\ \framebox[\abc]{жираф}
```

слон
жираф

При помощи перечисленных выше команд можно измерять размеры любых боксов (в частности, размеры рисунков), а не только строчек текста. Напомним, что для LaTeX'а не имеет значения содержание бокса. Обозначение lr-text, которое мы используем в определениях команд, всего лишь указывает, что аргумент команды обрабатывается в строковой моде, но не обязательно должен быть последовательностью букв.

9.1.2. Поднятие и опускание бокса

Команда

⚠ `\raisebox{len}[height][depth]{lr-text}`

форматирует текст её последнего аргумента lr-text в строковой моде и поднимает его на расстояние len. Отрицательное значение длины len приводит к опусканию бокса:

```
Здесь \raisebox{.5\height}{\em поднимаем\/},
а здесь \raisebox{-.5\height}{\em опускаем\/}.
```

Здесь *поднимаем*,
а здесь *опускаем*.

Если у команды `\raisebox` указаны опции height и depth, они заставляют компилятор думать, будто текст в боксе простирается на расстояние height вверх и на расстояние depth вниз от базисной линии строки. Например, команда

```
\raisebox{.4ex}[1.5ex][.75ex]{\scshape Abc}
```

не только поднимает слово Abc на высоту 0,4 ex, но также предлагает компилятору думать, будто текст простирается на 1,5 ex вверх над базисной линией и на 0,75 ex вниз от неё. При отсутствии опций height и depth компилятор использует реальные размеры строки. Напомним, что если указана только одна опция, то компилятор будет считать, что пропущена вторая опция, то есть указана только высота строки height.

Изменяя видимую высоту текста, можно регулировать интервал между строками. Например, опции height и/или depth позволяют иногда удалить лишнее

пустое пространство над и/или под математическими формулами. В следующем примере команда \fbox делает видимым изменение высоты и глубины текста:

```
\fbox{\scshape abc}\\
\fbox{\raisebox{0ex}[2ex][1ex]{\scshape abc}}
```

ABC

ABC

В совокупности с \framebox команда \raisebox позволяет регулировать как горизонтальные, так и вертикальные размеры рамки. Ещё один способ изменения вертикальных размеров боксов предоставляет команда \rule (раздел 9.3).

9.1.3. Сохранение бокса

Если какой-то фрагмент текста многократно повторяется, можно создать новую команду с помощью \newcommand (раздел 7.1), которая бы воспроизводила этот текст всякий раз, когда LaTeX встретит её во входном файле. Такой путь, сокращая время подготовки входного файла, не эффективен в смысле затрат времени на его обработку, так как LaTeX будет форматировать этот текст каждый раз заново. Если текст достаточно сложный (например, содержит много команд рисования), его обработка может заметно замедлиться. Однако LaTeX предоставляет возможность быстро распечатать однажды сформатированный бокс, имеющий определённое имя. Имя такого бокса декларируется командой

⚠ \newsavebox{*cmd*}

где *cmd* есть имя команды, не определённой ранее. Вновь определённая команда *cmd* имеет глобальную область действия, которая не ограничивается ни фигурными, ни командными скобками.

После того как имя бокса объявлено, любая из команд

⚠ \savebox{*cmd*}[*width*][*hpos*]{*lr-text*}
\sbox{*cmd*}{*lr-text*}

создаёт поименованный бокс, форматируя его содержимое в строковой моде, как это делают команды \makebox и \mbox. Однако, в отличие от последних, результат форматирования не печатается, а запоминается под указанным именем *cmd*. Опции *width* и *hpos* у команды \savebox имеют то же назначение, что и у команды \makebox: *width* есть ширина бокса, а опция *hpos* может принимать значения l, c, r или s соответственно тому, как следует позиционировать текст в боксе. В отличие от \newsavebox область действия команд \savebox и \sbox определяется обычными правилами.

Процедура

\begin{lrbox}{*cmd*} *lr-text* \end{lrbox}

действует аналогично команде \sbox. Сама процедура lrbox ничего не печатает, она запоминает сформатированный текст *lr-text* в виде команды *cmd*. В этом она эквивалентна \sbox{*cmd*}{*lr-text*} за тем исключением, что любые пробелы в

начале или конце *lr-text* игнорируются. Кроме того, в *lr-text* можно использовать команду `\verb` и процедуру `verbatim`, чего нельзя делать в аргументах команд.

Поименованный бокс печатает команда

⚠ `\usebox{cmd}`

Команда `\sbox` является аббревиатурой команды `\savebox` с пропущенными необязательными аргументами. Поэтому в следующем примере обе команды действуют совершенно одинаково:

```
\newsavebox{\prn}\savebox{\prn}{\scshape снова}
В тексте воспроизводится бокс \usebox{\prn},
\usebox{\prn} и \usebox{\prn}.
\sbox{\prn}{\em опять\/}%
Или \usebox{\prn} и \usebox{\prn}.
```

В тексте воспроизводится бокс СНОВА, СНОВА и СНОВА. Или *опять* и *опять*.

Здесь бокс с именем `\prn` определяется как логос, который сначала печатает слово «СНОВА», а затем слово «*опять*».

9.1.4. Параметры настройки

`\fboxrule` — толщина линий рамки, которую рисуют команды `\framebox` и `\fbox`. Однако для графической версии команды `\framebox`, используемой в процедуре `picture`, толщина линий рамки определяется другими декларациями, общими для всех команд в графической моде (раздел 9.5).

`\fboxsep` — ширина промежутка между рамкой и текстом в боксе, создаваемым командами `\framebox` и `\fbox`. Не действует для команды `\framebox` в графической моде.

9.2. Текстовые боксы

Парбокс — это бокс, содержимое которого печатается в текстовой моде, когда LaTeX'у разрешено разбивать текст на строки. Например, процедуры `figure` и `table` (глава 11) формируют парбоксы. Их ширина равна ширине окружающего текста, а сами боксы «плавают», т. е. могут переходить в такое место печатного документа, где бы они не выходили за нижнюю или верхнюю границу страницы. Однако LaTeX позволяет также формировать парбоксы заданной ширины и вставлять их в заданной точке текста при помощи команды `\parbox`:

⚠ `\parbox[vpos][height][inner-vpos]{width}{text}`

Она обычно используется для того, чтобы вставить один или несколько абзацев в рисунок или таблицу. LaTeX форматирует *text* в текстовой моде в виде последовательности строк ширины *width*. Аналогично случаю строковых боксов, команды `\height`, `\depth`, `\totalheight` и `\width` могут использоваться в аргументе *height*, где они обозначают естественные размеры бокса.

Опция *vpos* конкретизирует способ позиционирования бокса по вертикали относительно текущей строки:

`t` — базисная линия верхней строки в боксе выравнивается по базисной линии текущей строки;

`c` — центр бокса выравнивается по центру текущей строки (используется по умолчанию);

`b` — базисная линия нижней строки в боксе выравнивается по базисной линии текущей строки.

Опция *inner-vpos* действует как вертикальный эквивалент аргумента *vpos* в командах строковых боксов типа `\makebox`, фиксируя позиционирование содержимого *text* внутри бокса. Опция *inner-vpos* может принимать одно из четырёх значений `t`, `b`, `c` или `s`, соответственно означающих выравнивание вверх, вниз, по центру бокса или растяжение на его полную высоту за счёт изменения растяжимых вертикальных длин (если таковые имеются). Когда опция *inner-vpos* не указана, LaTeX приписывает ей то же значение, которое имеет опция *vpos*.

Перейдём к примерам. В первом из них в одной строке создаются два парбокса с опциями позиционирования `t` и `b`:

```
\parbox[t]{3.5cm}{Левый бокс расположен ниже строки.}
\ Это текущая строка. \
\parbox[b]{3.5cm}{Правый бокс расположен выше строки.}
```

В печатном документе эти боксы выравниваются соответственно по первой и последней строкам:

⟵ 3,5 см ⟶		Правый бокс расположен выше строки.
Левый бокс расположен ниже строки.	Это текущая строка.	⟵ 3,5 см ⟶

Внутри парбокса, образуемого командой `\parbox`, абзацный отступ отсутствует, так как его длина `\parindent` (раздел 17.2) по умолчанию там равна нулю. Однако значение `\parindent` можно изменить при помощи команды `\setlength` (раздел 2.10).

В узких парбоксах могут образоваться большие пробелы между словами, поскольку бывает трудно найти подходящее место для переноса слов по слогам. В этом случае часто отменяют режим выравнивания строк по правой границе (раздел 4.4).

```
Пример: \
\parbox[t]{1.3in}{\raggedright
в этом парбоксе использовано
 выравнивание текста влево.}
```

Пример: в этом парбоксе использовано выравнивание текста влево.

Усложним пример, использовав весь набор опций, чтобы сделать бокс с высотой в 40% ширины, растянув текст в нём на полную высоту.

```
Пример: \
\fbox{\parbox[t][.4\width][s]{1.3in}
{\raggedright этот парбокс
\smallskip растянут на полную высоту.}}
```

Пример: этот парбокс растянут на полную

высоту.

Здесь растяжимую длину вносит команда `\smallskip`; она срабатывает после завершения текущей строки. Для наглядности парбокс обведён рамкой.

Обычно команда `\parbox` используется для создания боксов с небольшими фрагментами текста. Она имеет ограниченную область применимости, так как внутри формируемого ею парбокса нельзя использовать процедуры составления списков (глава 5), процедуру `tabular` для составления таблиц (глава 12), а также команды `\footnote` и `\footnotetext`, которые печатают подстрочные примечания.

Для создания больших парбоксов с таблицами, списками, подстрочными примечаниями и прочими украшениями используется процедура `minipage` (министраница):

```
\begin{minipage}[vpos][height] [inner-vpos]{width}
  text
\end{minipage}
```

Она имеет те же аргументы, что и команда `\parbox`, но в её теле *text* (в отличие от `\parbox`) могут появляться любые команды и процедуры, кроме плавающих объектов (глава 11).

Когда команда `\footnote` применяется в `minipage`, она печатает подстрочное примечание на «дне» парбокса, создаваемого этой процедурой. Это особенно полезно для примечаний внутри таблиц или рисунков. Более того, в отличие от обычной текстовой моды в теле процедуры `minipage` команда `\footnote` может появляться в любом месте, даже внутри другого бокса или внутри ячейки таблицы (глава 12). Чтобы сделать подстрочное примечание к объекту в `minipage` на обычном месте внизу страницы, нужно использовать команды `\footnotemark` и `\footnotetext`. Оба варианта размещения подстрочного примечания демонстрирует следующий пример.

```
\begin{minipage}[t]{55mm}
  Пример\footnote{Сноска внутри министраницы.} использования
  команды \verb|\footnote| в министранице.
\end{minipage}               \hfill И \hfill
\begin{minipage}[t]{55mm}
  Пример\footnotemark{} сноски внизу страницы.
\end{minipage}\footnotetext{Сноска внизу страницы.}
```

Здесь образуются два бокса с подстрочными примечаниями к слову «Пример»:

Пример[a] использования команды `\footnote` в министранице.	И	Пример[3] сноски внизу страницы.

[a] Сноска внутри министраницы.

Если одна министраница вложена в другую, то может случиться, что подстрочное примечание будет напечатано в нижней части другого парбокса.

[3] Сноска внизу страницы.

Выбор ширины парбокса — задача из области визуального проектирования печатного документа. Последовательное проведение принципа логического проектирования требует, чтобы LaTeX сам умел определять ширину парбокса. Обычно это невозможно, так как компилятор должен знать длину строк, на которую требуется разбить текст. Однако при форматировании тела процедуры `tabbing` (раздел 12.1) длина строки заранее не фиксирована, поскольку она определяется исходным текстом. Поэтому компилятор может установить ширину парбокса меньше указанной в аргументе *width*, если министраница состоит только из единственной процедуры `tabbing`, а самая длинная строка в `tabbing` короче, чем *width*. В последнем случае ширина парбокса будет равна длине этой строки.

В заключение этого раздела приведём достаточно сложный пример, показывающий, как определить новую процедуру (назовём её `fmpage`) для печати текстового бокса в рамке. Желаемая цель достигается в два шага. Сначала текст, идущий в бокс, запоминается в виде какой-нибудь команды (назовём её `\fmbox`). Для этого используем процедуру `lrbox`, в теле которой вызываем процедуру `minipage`. Затем сформатированную на первом шаге министраницу вызываем при помощи команды `\usebox`, которую помещаем в аргумент команды `\fbox`. В итоге имеем следующее определение новой процедуры:

```
\newsavebox{\fmbox}
\newenvironment{fmpage}[1]
   {\begin{lrbox}{\fmbox}\begin{minipage}{#1}}
   {\end{minipage}\end{lrbox}\fbox{\usebox{\fmbox}}}
```

С её помощью удаётся поместить в рамку даже текст, форматируемый процедурой `verbatim`:

```
\begin{fmpage}{0.82\textwidth}
\begin{verbatim}
  @  #  $  %  ^  & \_  {  }
 \@ \# \$ \% \^ \& \_ \{ \}
\end{verbatim}
\end{fmpage}
```

```
  @  #  $  %  ^  & \_  {  }
 \@ \# \$ \% \^ \& \_ \{ \}
```

Обычными средствами этого сделать не удаётся, так как процедуру `verbatim` нельзя помещать в аргументы других команд (раздел 5.5). В последнем примере `\textwidth` означает ширину текущей колонки текста (раздел 17.2).

9.3. Линейные боксы

Линейный бокс — это чёрный прямоугольник, который печатает команда

⚠ `\rule[`*vpos*`]{`*width*`}{`*height*`}`

где *vpos* определяет величину смещения бокса над базисной линией строки. Отрицательное значение *vpos* смещает бокс вниз. По умолчанию смещение равно нулю. Все аргументы команды — нерастяжимые длины.

```
Rule 1: \rule{1mm}{5mm}\\
Rule 2: \rule[4mm]{10mm}{2mm}
```

Rule 1: ▌ Rule 2: ▬

Если выбрать линейный бокс достаточно тонким, то можно рисовать вертикальные и горизонтальные линии.

```
Вертикальная: \rule{0.4pt}{5mm}\\
Горизонтальная: \rule{10mm}{.4pt}
```

Вертикальная: |
Горизонтальная: ———

Линейный бокс нулевой толщины называется *стратой*. Хотя страта невидима, тем не менее для неё отводится место.

```
Сравните \fbox{этот бокс}
с \fbox{\rule[-5mm]{0mm}{1cm}этим}.
```

Сравните этот бокс с этим.

Страта даёт возможность вставлять вертикальные пробелы там, где команда `\vspace` не может быть использована, например в математических формулах.

9.4. Рисунки

LaTeX имеет встроенные средства для создания несложных схем или чертежей, а также позволяет импортировать графические изображения, подготовленные специализированными графическими программами. Первая возможность реализуется процедурой `picture`, вторая — командой `\includegraphics` из графических пакетов graphics и graphicx. Коллекцию графических пакетов мы рассмотрим в главе 10. Читатель может сразу перейти к этой главе, пропустив остаток текущей. После освоения методов импортирования графических изображений нашему Читателю, вероятно, придётся обратиться к главе 11, чтобы познакомиться с тем, как LaTeX выбирает место для размещения больших рисунков и делает к ним подписи.

Собственные графические возможности LaTeX'а более чем ограниченны. В наше время вряд ли найдутся любители скрупулёзного вычисления координат начала и конца каждой линии, как того требует LaTeX. Пользователям можно рекомендовать для подготовки рисунков одну из графических программ[4], которые умеют описывать рисунки, построенные на экране дисплея, в виде последовательности команд LaTeX'а. Среди таких программ встречаются достаточно мощные, предоставляющие пользователю весьма комфортные условия работы. Подготовленный файл с описанием рисунка может быть переписан в нужное место входного файла или введён в него командой `\input`.

Процедура `picture` переводит LaTeX в специальную графическую моду и формирует бокс заданных размеров. В графической моде, строго говоря, кроме деклараций можно использовать только две команды: `\put` и `\multiput`, которые

[4] Например, `texcad`, `latexcad` или `Gnuplot`.

размещают графические объекты в точке с заданными координатами. При необходимости скомбинировать несколько графических изображений, импортированных посредством `\includegraphics`, процедура `picture` также может быть полезной. Она незаменима, если на импортированные рисунки необходимо наложить, например, математические формулы. Чтобы сделать нечто подобное, Читателю необходимо прочитать следующий раздел, где рассказано о системе координат, которую вводит процедура `picture`.

9.5. Процедура picture

В процедуре `picture` любой *графический объект*, будь то текст, отрезок прямой линии, вектор, окружность или рисунок, созданный другой процедурой `picture`, позиционируется посредством задания координат (*x*,*y*) этого объекта. Координаты в процедуре `picture` измеряются в единицах

```
\unitlength
```

В отличие от других параметров длины, графические координаты записываются только цифрами без указания единицы измерения (она всегда равна `\unitlength`). Так, запись `(-0.5,1.75)` означает, что координата x равна `-0.5\unitlength`, а координата y равна `1.75\unitlength`. По умолчанию за единицу измерения принят 1 пункт, т. е. `\unitlength=1pt`. Чтобы изменить значение `\unitlength`, можно использовать команду `\setlength` (раздел 2.10). Переопределение `\unitlength` изменяет физические размеры сразу всех графических объектов в рисунках, то есть длину всех линий и диаметры всех окружностей. Однако толщина линий и размеры шрифтов не зависят от `\unitlength`, и, следовательно, полностью пропорции рисунка не сохраняются. Область действия `\unitlength` подчиняется обычным правилам, поэтому в разных рисунках можно использовать разные значения `\unitlength`. При этом важно, что переопределение `\unitlength` внутри рисунка (то есть внутри тела процедуры `picture`) запрещено, так как может привести к непредсказуемым последствиям. Об одном исключении из этого запрета мы расскажем в конце раздела 9.5.1. Процедура `picture` создаёт бокс указываемого размера с началом координат в левом нижнем углу, как показано на рис. 9.3. Полный вариант процедуры `picture` имеет две пары аргументов:

```
\begin{picture}(width,height)(x0,y0)
  pict-cmds
\end{picture}
```

Первая пара аргументов (*width*,*height*) обязательна, тогда как вторая пара (x_0,y_0) может отсутствовать. В отличие от обычных процедур, и обязательный, и необязательный аргументы записываются в круглых скобках, чтобы подчеркнуть, что они являются длинами, которые измеряются в единицах `\unitlength`. Процедура `picture` строит графический бокс. Ширина и высота графического бокса определяется парой неотрицательных чисел (*width*,*height*) в первой (обязательной) паре аргументов процедуры. Первый аргумент даёт указание LaTeX'у зарезервировать свободное пространство указанного размера для размещения графического бокса. Однако LaTeX не проверяет, выходят или нет графические объекты за пределы бокса. Поэтому можно рисовать за границами бокса и даже за границами страницы.

Если вторая (необязательная) пара аргументов (x_0,y_0) отсутствует, то начало координат помещается в левый нижний угол графического бокса и ему присваиваются координаты $(0, 0)$.

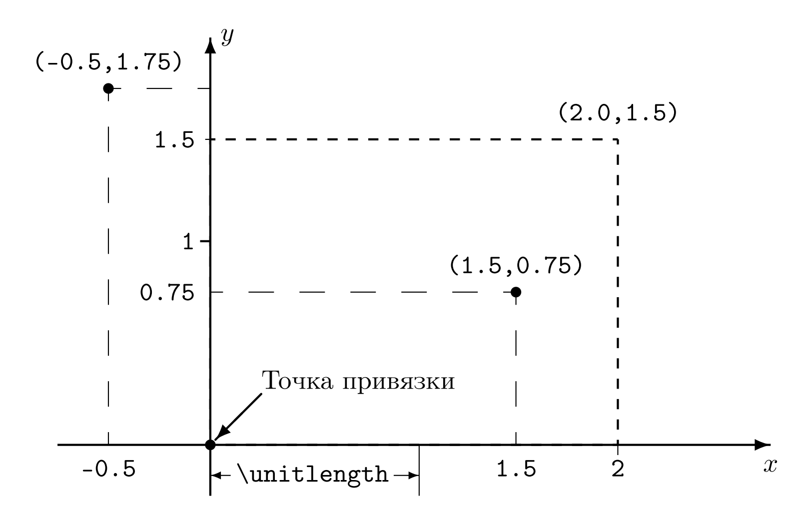


Рис. 9.3. Система координат в графическом боксе, введённая командой `\begin{picture}` `(2.0,1.5)`. Коротким пунктиром показаны границы графического бокса

В противном случае координаты левого нижнего угла приравниваются значениям (x_0,y_0). Например, команда

```
\begin{picture}(50,70)(30,-20)
```

производит бокс шириной 50 и высотой 70 единиц `\unitlength`, левый нижний угол которого имеет координаты $(30, -20)$. Тогда координаты правого верхнего угла будут равны $(80, 50)$. На начальной стадии подготовки рисунка начало координат обычно помещают в левый нижний угол, опуская необязательную пару аргументов. Позднее, если необходимо сдвинуть весь рисунок целиком, это можно сделать, введя соответствующие значения x_0 и y_0. Графические программы редактирования рисунков для LaTeX'а действуют более прямолинейно: они проводят сдвиг рисунка путём перерасчёта координат всех графических объектов.

Команда `\begin{picture}` переводит LaTeX в специальную графическую моду, в которой помимо деклараций могут использоваться следующие команды:

```
\put    \multiput    \qbezier    \graphpaper
```

Другие команды разрешены только в аргументах команд позиционирования `\put` и `\multiput`.

LaTeX обеспечивает две стандартные толщины линий для рисунков: тонкие, как на этом овале ⬭, или толстые, как на этом ⬭. Переключение между толстыми и тонкими линиями осуществляют декларации

```
\thinlines
\thicklines
```

По умолчанию принята `\thinlines` (тонкие линии). Декларация

```
\linethickness{len}
```

устанавливает, что толщина горизонтальных и вертикальных линий равна величине *len*, выраженной в любых единицах измерения длины. Например, декларация `\thinlines` эквивалентна `\linethickness{0.4pt}`. Однако `\linethickness` не изменяет толщину наклонных линий, а

также окружностей и закруглений в углах овалов. Все три перечисленные декларации являются устойчивыми, они могут появляться в любом месте входного файла. Область их действия подчиняется обычным правилам, и они могут быть использованы в любое время.

Ниже во всех примерах данной главы, иллюстрирующих действие команд рисования, предполагается, что все команды выполняются в графической моде, но команды `\begin{picture}` и `\end{picture}` могут быть опущены для упрощения записи. Мы используем различные значения `\unitlength`, не указывая их явно. Результат действия иллюстрируемых команд изображается толстыми линиями при задействованной декларации `\thicklines`, тогда как вспомогательные элементы рисунков, такие как стрелки, направленные в точки привязки, нарисованы тонкими линиями при задействованной декларации `\thinlines`.

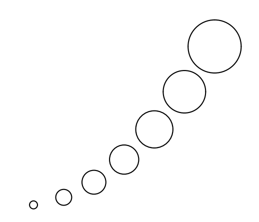
Если процедура `picture` производит большой бокс, его нужно размещать как плавающий объект (глава 11). Все рисующие команды хрупкие, поэтому при необходимости их следует защищать командой `\protect` (раздел 2.7).

9.5.1. Команды позиционирования

```
⚠ \put(x,y){pict-obj}
⚠ \multiput(x,y)(dx,dy){n}{pict-obj}
```

Команда `\put` помещает графический объект pict-obj так, чтобы его *точка привязки* (reference point) находилась в точке с координатами (x,y). Графическим объектом может быть любой текст. LaTeX форматирует этот текст в строковой моде, причём точкой привязки является левый нижний угол строкового бокса или другая точка, если графический объект создаёт одна из команд рисования, описанная в следующих разделах данной главы. Команда `\multiput` создаёт n копий графического объекта pict-obj, размещая их так, чтобы точка привязки j-ой копии попадала в точку с координатами $(x + [j - 1]dx, y + [j - 1]dy)$. Первая копия привязывается к точке (x,y), вторая — к точке $(x + dx, y + dy)$, последняя — к точке $(x + [n - 1]dx, y + [n - 1]dy)$.

```
\newcounter{N}\setcounter{N}{0}
\setlength{\unitlength}{1mm}
\begin{picture}(45,25)
  \multiput(3,3)(4,\value{N}){7}
  {\addtocounter{N}{1}%
  \circle{\value{N}}}
\end{picture}
```



Как показывает этот пример, форматирование объектов осуществляется каждый раз заново, поэтому они могут различаться. Более того, выбрав переменное значение dy, мы расположили эти копии на разном расстоянии друг от друга.

Поместив одну команду `\multiput` в аргумент другой такой же команды, можно создавать двумерные рисунки из повторяющихся элементов. Так как каждую копию рисунка `\multiput` форматирует заново, «внутреннюю» команду можно запомнить в поименованном боксе, чтобы ускорить работу компилятора.

Ещё одно замечание относится к командной длине `\unitlength`. Мы уже отмечали, что её нельзя переопределять внутри процедуры `picture`. Однако рисунок может содержать вложенные рисунки. Выделение в рисунке рисунка поменьше вообще является ценной идеей. Вложенный рисунок создаёт дополнительная процедура `picture`, вставленная в аргумент команды `\put`:

```
\put(17.67,14.13){\begin{picture}(10,12) ... \end{picture}}
```

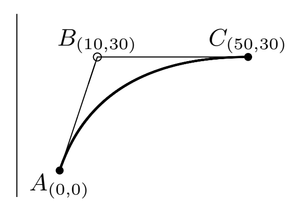
Таким способом легко переставлять кусочки рисунка относительно друг друга. Размер таких кусочков также может быть изменён, если переопределить значение `\unitlength` непосредственно в аргументе команды `\put`:

```
\put(1,3){\setlength{\unitlength}{1in}%
  \begin{picture}(1,2)
      ...
  \end{picture}}
```

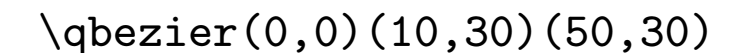
9.5.2. Кривые Безье

LaTeX позволяет конструировать достаточно сложные математические кривые, используя технику аппроксимации сплайнами Безье. Эта техника была заимствована из языка описания страниц PostScript, где также используется аппроксимация сплайнами Безье как базис для рисования кривых линий. Для построения квадратичной кривой Безье требуется задать на плоскости координаты трёх точек: двух концевых и одной контрольной. Квадратичная кривая Безье является квадратичной параболой, проходящей через заданные концевые точки так, что прямые, соединяющие концевые точки с контрольной, касаются параболы в концевых точках. В следующем примере концевые точки обозначены буквами A и C, а контрольная точка — буквой B.

```
\qbezier(0,0)(10,30)(50,30)
```



Полный синтаксис команды


`\qbezier[`num`](`x_A`,`y_A`)(`x_B`,`y_B`)(`x_C`,`y_C`)`

помимо координат точек A, B, C содержит ещё один необязательный аргумент num, который задаёт число точек на кривой. Если опция num опущена, рисуется гладкая линия с максимальным количеством точек, которое определяется значением

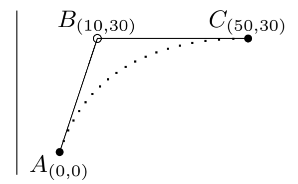
`\qbeziermax`

Это значение можно изменить при помощи `\renewcommand`. Например,

```
\renewcommand{\qbeziermax}{250}
```

Второй пример показывает, к чему приводит добавление опции num:

```
\qbezier[20](0,0)(10,30)(50,30)
```



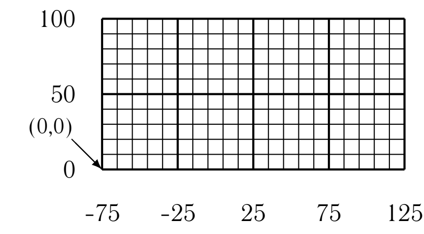
9.5.3. Сетка

Пакет **graphpap** определяет команду

`\graphpaper[`*spacing*`](`x_0`,`y_0`)(`*x-dimen*`,`*y-dimen*`)` (graphpap)

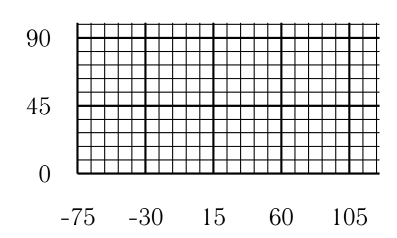
которая рисует пронумерованную координатную сетку. Первый аргумент (x_0,y_0) задаёт координаты левого нижнего угла сетки, а второй аргумент (*x-dimen*,*y-dimen*) задаёт ширину и высоту сетки.

```
\graphpaper(-75,0)(200,100)
```



Каждая координатная линия проводится через 10 единиц (выраженных в `\unitlength`). Необязательный первый аргумент *spacing* позволяет изменить это правило.

```
\graphpaper[9](-75,0)(200,100)
```



Все аргументы `\graphpaper` должны быть целыми числами.

9.6. Графические объекты

Графическим объектом становится любой текст, если его поместить в последний аргумент *pict-obj* команды `\put` или `\multiput`. LaTeX форматирует его в строковой моде, а точку привязки помещает в нижний левый угол строкового бокса. Простейшим примером служит обычный текст:

```
\put(12,2){\large\bfseries указ}
```

указ

Внимательный Читатель, должно быть, заметил, что нижний угол лежит на базисной линии строки, а буква «**у**» простирается ниже.

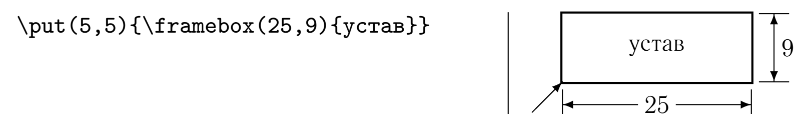
9.6.1. Боксы в рисунках

От обычных строковых боксов, описанных в начале главы, графические боксы отличаются прежде всего тем, что необходимо задать как ширину, так и высоту боксов.

⚠ `\makebox(`*width*`,`*height*`)[`*hvpos*`]{`*lr-text*`}`
⚠ `\framebox(`*width*`,`*height*`)[`*hvpos*`]{`*lr-text*`}`
⚠ `\dashbox{`*dash*`}(`*width*`,`*height*`)[`*hvpos*`]{`*lr-text*`}`
⚠ `\frame{`*lr-text*`}`

Пара аргументов (*width*,*height*) задаёт соответственно ширину и высоту бокса в единицах длины `\unitlength`. Точка привязки находится в левом нижнем углу бокса.

```
\put(5,5){\framebox(25,9){устав}}
```



Опция *hvpos* управляет позиционированием текста *lr-text* внутри бокса. Она может состоять из одного или двух спецификаторов, определяющих горизонтальное и вертикальное позиционирование:

- `l` — горизонтальный сдвиг к левому краю бокса,
- `r` — горизонтальный сдвиг к правому краю бокса,
- `s` — растяжение текста на всю ширину бокса,
- `t` — вертикальный сдвиг к верхнему краю бокса,
- `b` — вертикальный сдвиг к нижнему краю бокса.

Порядок следования спецификаторов в опции *hvpos* не имеет значения, то есть `lt` равнозначно `tl`. Допустим также спецификатор `c`, означающий центрирование текста по вертикали и/или горизонтали, но он действует по умолчанию. Команда `\framebox` по краям бокса проводит сплошные линии, а команда `\dashbox` — пунктирные, где длина штриха должна быть задана положительным числом *dash* в единицах `\unitlength`. Для наилучшего результата размеры бокса *width* и *height* должны быть кратны параметру *dash*. В отличие от текстового варианта `\framebox` графические команды не вставляют пробелы между рамкой и текстом:

```
\dashbox{4}(24,12)[sb]{Указ №1}}
```

Указ №1

Хотя в графической моде можно применять только команды, формирующие графические боксы, сами эти команды могут быть использованы в любом окружении, даже вне процедуры `picture`.

Команда `\makebox(0,0){...}`, формирующая бокс нулевого размера, часто бывает полезна для позиционирования текста в рисунках. Она помещает точку привязки в центр текста, обозначенного здесь многоточием. Следующий пример показывает, как можно позиционировать текст, варьируя расположение точки привязки:

```
\put(10,0){\makebox(0,0){ухо}}
\put(22,0){\makebox(0,0)[b]{горло}}
\put(34,0){\makebox(0,0)[tr]{нос}}
```

ухо горло нос

Команда `\fbox` (раздел 9.1) чертит рамку вокруг текста, оставляя некоторый зазор между рамкой и текстом. Команда `\frame{`*lr-text*`}` работает подобным образом, но не оставляет зазора между текстом и рамкой. В отличие от графического варианта команды `\framebox` (см. выше), который рисует рамку заданных размеров, команда `\frame` размеры рамки выбирает сама.

```
\put(5,5){\frame{УХА}}
\put(25,5){\fbox{УХА}}
```



9.6.2. Прямые линии

Прямые линии в процедуре `picture` имеют конечный, хотя и широкий набор углов наклона к осям рисунка. Поэтому линию нельзя задать, указывая её начальную и конечную точки: в этом случае в репертуаре LaTeX'а может не найтись нужного наклона. Способ описания линии, принятый LaTeX'ом, очень прост, но всё-таки нуждается в пояснении. Линию рисует команда

⚠ `\line(`x_s`,`y_s`){`dx`}`

где длины x_s, y_s, выраженные в единицах `\unitlength`, определяют угол наклона линии. Если `(`x`,`y`)` — точка, из которой проводится линия (она устанавливается командой `\put(`x`,`y`)`), то вторая (прицельная) точка имеет координаты `(`x+x_s`,`y+y_s`)`, как показано на рисунке 9.4.

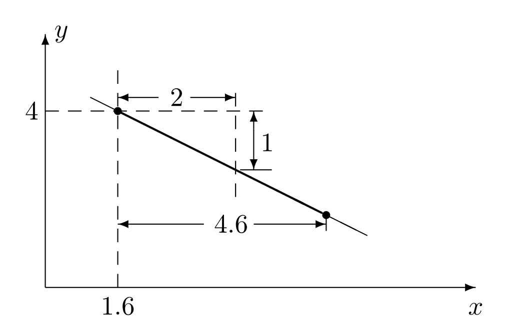


Рис. 9.4. Построение наклонной линии в графическом боксе на примере команды `\put(1.6,4){\line(2,-1){4.6}}`

LaTeX проводит линию из начальной точки через прицельную до точки, имеющей координату (x+dx). Если $x_s = 0$, то линия проводится из начальной точки вертикально вверх или вниз (в зависимости от знака y_s) на расстояние dx (в единицах `\unitlength`), которое должно быть положительным. Числа x_s и y_s должны быть целыми, лежать в диапазоне от -6 до $+6$ и не иметь общего делителя, отличного от единицы.

LaTeX рисует наклонные линии, набирая их из специальных символов, состоящих из кусочков прямой линии. Поэтому существует предельно малая длина наклонной линии (около 10 pt). При попытке нарисовать ещё более короткую линию LaTeX ничего не напечатает.

9.6.3. Стрелки

Команда построения стрелки

⚠ `\vector(`x_s`,`y_s`){`dx`}`

работает аналогично команде `\line`. Однако здесь величины x_s и y_s должны лежать в диапазоне от -4 до $+4$.

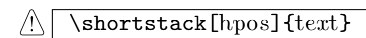
```
\put(4,5){\vector(1,2){5}}
\put(25,2){\vector(-1,0){7}}
\put(32,15){\vector(3,-2){12}}
```

(32,15)

(4,5) (25,2)

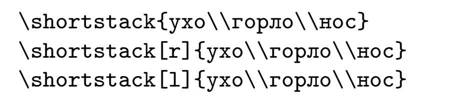
9.6.4. Стеки

Команда

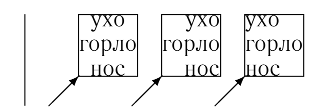


```
\shortstack[hpos]{text}
```

производит бокс, в котором текст располагается в одну колонку. Точка привязки находится в нижнем левом углу бокса. Обязательный аргумент *text* содержит текст, разделённый на строки командами `\\`. Команда `\shortstack` не оставляет пробелов по краям строк. По умолчанию все слова в боксе центрируются, но их можно сдвинуть влево или вправо, задав в опции *hpos* спецификатор `l` или `r`; допускаются также спецификаторы `s` и `c`. В приведённом ниже примере стеки для лучшего понимания обведены рамкой, а команда `\put`, устанавливающая точки привязки на кончики стрелок, опущена.



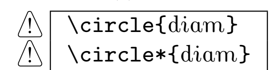
```
\shortstack{ухо\\горло\\нос}
\shortstack[r]{ухо\\горло\\нос}
\shortstack[l]{ухо\\горло\\нос}
```



Строки в стеке можно раздвинуть, используя команду `\\` с опцией (раздел 4.4) или вставляя страту (раздел 9.3). Хотя команда `\shortstack` формирует обычный бокс и работает в любой моде, её редко используют вне процедуры `picture`.

9.6.5. Круги

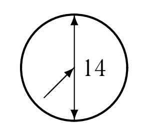
Команды

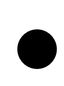


```
\circle{diam}
\circle*{diam}
```

рисуют соответственно окружность и круг. Диаметр *diam* измеряется в единицах `\unitlength`.

```
\put(30,10){\circle{14}}
\put(10,14){\circle*{6}}
```

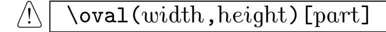




Точками привязки служат центры окружности и круга. LaTeX имеет дискретный набор окружностей и кругов, который определяется имеющимся набором шрифтов. Команды `\circle` и `\circle*` выбирают из этого набора наиболее близкие к указанному диаметру *diam*. Диаметр наибольшей окружности вряд ли значительно превышает 40 пунктов (примерно полдюйма), диаметр наибольшего круга примерно вдвое меньше.

9.6.6. Овалы и закруглённые углы

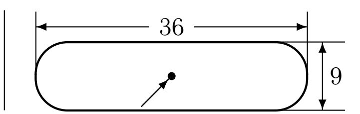
Овал — это прямоугольник, у которого углы заменены четвертями окружности наибольшего возможного радиуса. Овал рисует команда



```
\oval(width,height)[part]
```

Точка привязки находится в центре фигуры. Пара аргументов в круглых скобках задаёт ширину и высоту овала (в единицах `\unitlength`).

```
\put(20,8.5){\oval(36,9)}
```



По умолчанию LaTeX рисует полный овал. Добавление необязательного аргумента *part* позволяет нарисовать любую половину или четверть овала. Опция *part* может состоять из одного или двух спецификаторов: `r` (правый), `l` (левый), `t` (верхний), `b` (нижний). Один спецификатор соответствует половине овала (правой, левой, верхней или нижней), два спецификатора — четверти овала. Например, с опцией `bl` (или `lb`) команда `\oval` нарисует нижнюю левую четверть овала. Размеры и точка привязки определяются по полному овалу вне зависимости от наличия опции: *part* всего лишь подавляет рисование ненужных частей овала.

```
\put(12,10){\oval(28,8)[b]}
\put(43,10){\oval(17,8)[tl]}
```

(12,10)
8
(43,10)

Углы с малым радиусом закругления (меньшим, чем делает команда `\oval`) можно построить, соединяя прямые линии четвертями овала.

```
\put(5,8){\line(0,1){3.5}}
\put(8,8){\oval(6,6)[lb]}
\put(8,5){\vector(1,0){16}}
```

(8,8)

9.7. Копирование рисунка

Подобно `\makebox` и `\framebox`, команда `\savebox` из раздела 9.1.3 имеет графический вариант, в котором размер бокса указывается парой аргументов в круглых скобках:

⚠ `\savebox{`*cmd*`}(`*width*`,`*height*`)[`*hvpos*`]{`*lr-text*`}`

Здесь *cmd* — имя бокса, которое должно быть предварительно объявлено при помощи декларации `\newsavebox{`*cmd*`}`. Назначение других аргументов такое же, как в команде `\framebox`. Команда `\savebox` указывает LaTeX'у, что подготовленный ею бокс необходимо запомнить под именем *cmd*. Печатает поименованный бокс команда `\usebox{`*cmd*`}`. Она быстро воспроизводит бокс, не затрачивая времени на его форматирование. Поэтому рекомендуется сохранять графические объекты в виде боксов, если они появляются в нескольких рисунках или в нескольких местах одного и того же рисунка.

```
\savebox{\prn}(12,6)[tr]{\shortstack{У\\Х\\А}}
\put(5,5){\frame{\usebox{\prn}}}
\put(25,5){\frame{\usebox{\prn}}}
```

У Х А   У Х А

Команда `\sbox` в графической моде сохраняет свои обычные свойства.

```
\sbox{\prn}{\shortstack{У\\Х\\А}}
\put(5,5){\frame{\usebox{\prn}}}
\put(25,5){\fbox{\usebox{\prn}}}
```

У Х А   У Х А

На хранение поименованных боксов LaTeX расходует отведённую ему память. Поэтому не следует хранить содержимое поименованного бокса дольше, чем необходимо. В этой связи полезно обратить внимание, что `\savebox` есть декларация (так как сама она ничего не печатает), а её область действия подчиняется обычным правилам. В частности, если поименованный бокс `\prn` был сохранён командой `\savebox` внутри процедуры `picture`, то его определение затирается при выходе из этой процедуры. Если же `\prn` использовался в нескольких рисунках и, следовательно, был сохранён вне процедуры `picture`, то его можно затереть командой `\sbox{\prn}{}`.

К вершине было не взнестись очам,
А склон был много круче полуоси,
Секущей четверть круга пополам.
А. Данте. Божественная комедия

Глава 10

Графика и цвет

Встраивание графических изображений, созданных другими приложениями, обеспечивают пакеты из коллекции graphics, написанной Дэвидом Карлайлом (Carlisle, David). Помимо импортирования рисунков эти пакеты также обеспечивают работу с цветом, позволяют производить вращение или изменять масштаб любого бокса, будь то рисунок, таблица или отдельная буква.

При загрузке любого графического пакета нужно выбрать драйвер устройства, которое предполагается использовать для печати документа или, в более широком плане, для вывода документа на выходное устройство. Драйвер указывается в необязательном аргументе декларации `\usepackage` в преамбуле входного файла. Например,

```
\usepackage[dvips]{graphicx,color}
```

загружает пакеты graphicx и color с драйвером `dvips`, а

```
\usepackage[pdftex]{graphicx,color}
```

предполагает использование драйвера `pdftex`. В данном контексте драйвером называется опция, определяющая набор инструкций для преобразования команд LaTeX'а в команды, которые управляют работой программы-драйвера конкретного выходного устройства. Загружая графический пакет, декларация `\usepackage` производит необходимые настройки для последующей работы этой программы, но не запускает её.

Как мы отмечали в разделе 1.14, компилятор `latex` из исходного файла с именем *jobname*`.tex` создаёт `dvi`-файл *jobname*`.dvi`, который затем при помощи драйвера выходного устройства можно распечатать на принтере или вывести на экран монитора. Расширение `dvi` имени файла происходит от слов device independent file, что по замыслу Д. Кнута должно было означать «файл, не зависящий от устройства». Замысел не удалось реализовать в части, касающейся как раз графики и цвета, потому что невозможно было договориться заранее о единых стандартах в этой быстро развивающейся области информационных технологий. Однако время всё расставило по местам, выявив лидеров среди разнообразных графических форматов и программ-драйверов. Проверку временем выдержали язык описания страниц PostScript и программа `dvips` Томаша Рокицки (Rokicki, Tomas), которая преобразует `dvi`-файл в файл PostScript с расширением `ps`.

Программа dvips долгое время имела слишком очевидные преимущества перед другими приложениями, что и обеспечило её преобладающее распространение. Она единственная поддерживала все графические возможности пакетов из коллекции graphics. Соответственно только драйвер dvips был реализован для всех типов компьютеров и для всех операционных систем. Поэтому сейчас новые драйверы обычно обладают всеми возможностями dvips, позволяя загружать графические пакеты с опцией dvips.

Небольшой диссонанс внесло изобретение переносимого формата документов Portable Document Format, или просто PDF. С появлением компилятора pdflatex стала реальностью прямая компиляция документов с разметкой LaTeX в формат PDF, но в таком случае при загрузке графических пакетов следует указывать опцию pdftex вместо dvips.

Кстати, а что произойдёт, если при загрузке графического пакета драйвер вообще не указан? Вопрос вполне разумный, ведь драйвер указывается в необязательном аргументе! Ответ содержится в конфигурационных файлах, где указан драйвер, используемый по умолчанию. Конфигурационные файлы графических пакетов graphics.cfg и color.cfg хранятся в одном из каталогов, где компилятор ищет необходимую ему информацию. Графические пакеты сконфигурированы так, что компилятор latex выбирает опцию dvips, а компилятор pdflatex выбирает опцию pdftex. Таким образом, лучше всего вообще не указывать, какой именно нужен драйвер.

Подробнее мы обсудим тему драйверной поддержки в конце главы в разделе 10.9.1, а сейчас предположим, что выбор драйвера производится автоматически, т. е. графические пакеты загружены без явного указания драйвера:

```
\usepackage{graphicx,color}
```

Редкие случаи, когда есть хотя бы минимальное различие в использовании драйверов dvips или pdftex, используемых по умолчанию, мы будем оговаривать особо.

В этой главе мы расскажем, как в печатный документ вставить (импортировать) различного рода графические объекты: рисунки, графики, чертежи и пр. Мы также объясним, как изменить размеры, пропорции и цвет рисунка или как его повернуть. Читатель увидит, что подобные манипуляции возможны не только с рисунками, приготовленными специализированными графическими программами, но и вообще с любой составной частью печатного текста, будь то обычный текстовой бокс, математическая формула или таблица. Однако обо всём по порядку. . .

Основу коллекции graphics составляют пакеты graphics, graphicx, color, epsfig и lscape. Первый из них иногда называют стандартным графическим пакетом, а второй — расширенным или продвинутым. Эта два пакета вводят один и тот же набор команд, которые, однако, различаются способом задания необязательных аргументов. Пакет graphicx кажется более удобным, более дружественным для пользователя, чем graphics, но, возможно, это мнение лишь отражает наши пристрастия, а не объективный факт.

Пакет color предназначен для работы с цветом.

Пакет epsfig входит в коллекцию для совместимости с устаревшей версией, сделанной в своё время для формата LaTeX 2.09, а пакет lscape используют для печати отдельных страниц в альбомной (горизонтальной) ориентации, вместо более привычной портретной (вертикальной).

Дальнейшее изложение мы построим по функциональному признаку и посвятим отдельные разделы вращению боксов, их масштабированию, импортированию рисунков и работе с цветом. Читатель без труда определит, к какому пакету принадлежит та или иная команда, так как название пакета указывается в скобках справа от синтаксического описания каждой команды. Если Читателя интересует только проблема импортирования рисунка в печатный документ, он может обратиться сразу к разделу 10.3.

10.1. Вращение боксов

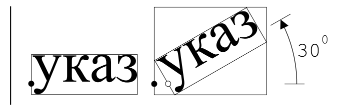
Команда

⚠ `\rotatebox{angle}{lr-text}` (graphics)

помещает текст *lr-text* в бокс, как это делает `\mbox` (глава 9), и поворачивает его на *angle* градусов против часовой стрелки. Полученный бокс есть наименьший прямоугольник (ориентированный стандартным образом), в который целиком помещается *lr-text*; его точка привязки находится с левой стороны на той же высоте, что и у повёрнутого бокса. В следующем примере точки привязки изображены маленькими кружками:



```
указ \rotatebox{30}{указ}
```



Текст *lr-text*, идущий в бокс, обрабатывается в строковой моде, но может иметь любую природу. В частности, он может содержать команду `\includegraphics`, которая импортирует графические файлы и о которой мы расскажем в разделе 10.3. С помощью `\rotatebox` удобно делать вертикальные надписи в столбцах таблиц, как в табл. 10.1 на стр. 264. Отрицательное значение угла *angle* соответствует вращению по часовой стрелке.

```
Вращаем \rotatebox{90}{против} и
\rotatebox{-90}{по} часовой стрелке.
```

Вращаем против и по часовой стрелке.

Если Читатель попытается повторить этот пример на своём компьютере, но увидит на экране монитора нечто вроде «Вращаем протипво часовой стрелке», ему придётся перечитать начало этой главы, а также обратиться к рисункам 1.2–1.7 на страницах 41–45, чтобы вспомнить, как работает LaTeX. Напомним, что

компилятор `latex`, обработав исходный файл *jobname*`.tex`, записывает «готовый документ» в `dvi`-файл *jobname*`.dvi` (рис. 1.2). Возможно, при просмотре этого файла на экране с помощью DVI-обозревателя, «повёрнутый» текст вовсе не будет повёрнут. Именно так и случится, если используется программа `YAP` из комплекта MiKTeX, так как она не поддерживает функцию вращения[1] (рис. 1.4). Дело в том, что в `dvi`-файле записаны инструкции для DVI-обозревателя, а не собственно изображение, поэтому результат зависит от «продвинутости» обозревателя. Однако какой-то более совершенный DVI-обозреватель покажет тот же `dvi`-файл правильно. Если выполнить компиляцию исходного текста в `pdf`-файл при помощи программы `pdflatex` (рис. 1.3) и просмотреть полученный файл *jobname*`.pdf` в Adobe Reader (рис. 1.5), то желанное вращение обязательно будет получено. Можно также экспортировать *jobname*`.dvi` в PostScript-файл при помощи программы `dvips` (рис. 1.6) и лишь затем просмотреть полученный таким образом *jobname*`.ps` при помощи программы GSview (рис. 1.7). Как правило, все операции по взаимодействию перечисленных программ выполняет редактор исходных файлов LaTeX. От пользователя только требуется нажимать подходящие клавиши. На всякий случай приведём полностью исходный текст входного файла:

```
\documentclass{article}
\usepackage{graphics}
\begin{document}
  Вращаем \rotatebox{90}{против} и \rotatebox{-90}{по} часовой стрелке.
\end{document}
```

Пакет graphicx вводит расширенную версию команды `\rotatebox`, которая имеет один необязательный аргумент.

⚠ `\rotatebox[`*keyval-list*`]{`*angle*`}{`*lr-text*`}` (graphicx)

По умолчанию объект вращается относительно точки привязки. Опция *keyval-list* позволяет изменить положение оси вращения, а также единицу измерения угла поворота. В необязательном аргументе *keyval-list* через запятую могут быть перечислены *ключи* следующего вида:

`origin=`*pos*
`x=`*dimen1*
`y=`*dimen2*
`units=`*number*

Ключ `origin=`*pos* задаёт положение оси вращения, причём значение ключа *pos* может состоять из одной или двух букв, фиксирующих положение оси вращения по горизонтали и вертикали. Горизонтальная координата оси вращения фиксируется одним из трёх спецификаторов: `l` (left, слева), `c` (center, центр), `r` (right, справа), в то время как вертикальная координата определяется одним из четырёх

[1] Однако `YAP` успешно поворачивает импортируемые рисунки.

спецификаторов: t (top, верх), c (center, центр), B (Baseline, базисная линия[2]), b (bottom, низ). Например, origin=rb помещает ось вращения в правый нижний угол бокса, origin=lt — в верхний левый угол. Комбинация cB задаёт середину отрезка базисной линии, проходящей через вращаемый бокс, а lc определяет среднюю по высоте точку с левой стороны бокса. Точке привязки соответствует lB. Всего таким способом можно задать 12 точек, как показано на рис. 10.1. Порядок следования спецификаторов в *pos* не имеет значения: br эквивалентно rb. Если в *pos* задан только один спецификатор, то вторым предполагается c, который представляет горизонтальное или вертикальное центрирование в дополнение к тому, что задаёт другой спецификатор.

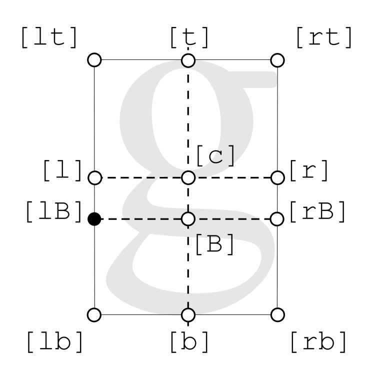


Рис. 10.1. Доступные точки вращения для ключа origin.
Точка привязки показана чёрным кружком

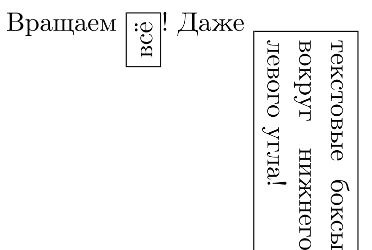
Если ось вращения не совпадает ни с одной из 12 точек, указанных на рис. 10.1, её можно задать с помощью ключей x=*dimen1* и y=*dimen2*, которые определяют расстояние от точки привязки бокса до оси вращения. Например, если [*keyval-list*] имеет вид [x=4mm,y=3mm], то бокс вращается вокруг точки, которая смещена относительно точки привязки на 4 мм по оси абсцисс и на 3 мм по оси ординат.

По умолчанию объект вращается относительно точки привязки, а угол *angle* измеряется в градусах. Ключ units=*number* позволяет выбирать единицы измерения угла вращения. Его значение *number* должно быть равно величине полного оборота в выбранных единицах. Например, units=360 означает, что угол будет измеряться в градусах, а units=6.283185 — в радианах.

Мы не будем комментировать последний пример в этом разделе. Просто предлагаем Читателю «почувствовать разницу». Отметим только, что этот пример сработает правильно при условии загрузки расширенной версии графического пакета graphicx.

[2] Напомним, что базисная линия проходит через точку привязки, а нижний край бокса находится на расстоянии \depth от неё (раздел 9.1).

```
Вращаем \rotatebox[origin=b]{90}{\fbox{всё}}!
Даже \rotatebox{-90}{\fbox{\parbox[b]{2.8cm}
{текстовые боксы вокруг нижнего левого угла!}}}
```



10.2. Масштабирование боксов

Размеры произвольного бокса и его содержимого можно изменить, проведя операцию масштабирования. Масштаб задают явно или неявно. В последнем случае реально задают размеры бокса, которые он должен иметь после масштабирования.

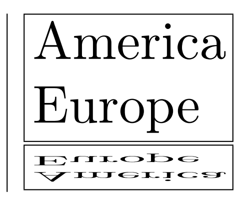
10.2.1. Масштабирование по заданному масштабу

Команда

⚠ `\scalebox{h-scale}[v-scale]{lr-text}` (graphics, graphicx)

масштабирует lr-text, увеличивая его ширину в h-scale раз, а высоту в v-scale раз. Отрицательные значения множителей приводят к отражению объекта относительно горизонтали и/или вертикали. Если опция [v-scale] пропущена, то её значение равно h-scale, при этом сохраняется аспектное отношение высоты бокса к его ширине.

```
\fbox{\scalebox{2}{\parbox{0.5in}
{America \\ Europe}}}
\fbox{\scalebox{2}[-0.5]{\parbox{0.5in}
{America \\ Europe}}}
```



10.2.2. Отражение бокса

Команда

⚠ `\reflectbox{lr-text}` (graphics, graphicx)

есть сокращение от `\scalebox{-1}[1]{lr-text}`.



```
$E=mc^2$ \reflectbox{$E=mc^2$}
```

$E = mc^2$

10.2.3. Масштабирование по заданному размеру

⚠ \resizebox{*width*}{*height*}{*lr-text*}
⚠ \resizebox*{*width*}{*totalheight*}{*lr-text*}

(graphics, graphicx)

Команда \resizebox подгоняет ширину и высоту бокса, в который помещается *lr-text*, к заданным величинам *width* и *height*, которые должны иметь размерность длины. Сохранить отношение высоты к ширине можно, указав в качестве *width* или *height* восклицательный знак !. Команда \resizebox* идентична \resizebox, но второй аргумент *totalheight* задаёт полную высоту бокса, а не высоту той его части, которая возвышается над базисной линией. Исходные размеры бокса (см. рис. 9.1 на стр. 210) доступны в аргументах *width*, *height* и *totalheight* в виде команд \height, \width, \totalheight, \depth так же, как в других командах, формирующих строковые боксы (раздел 9.1). Так, в следующем примере текст растягивается до ширины 2 дюйма без изменения первоначальной высоты бокса:

```
\resizebox{2in}{\height}{Нью Васюки}
```

Нью Васюки

Для сравнения покажем, что получается при сохранении аспектного отношения:

```
\resizebox{2in}{!}{Нью Васюки}
```

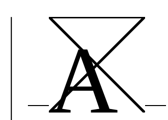
Нью Васюки

Качество масштабированного изображения зависит от того, является ли оно в своей основе растровым или векторным (см. 10.3.1). При сильном увеличении растровое изображение становится зернистым.

10.3. Импортирование графики

Импортирование изображений, созданных специализированными графическими программами, осуществляется с помощью команды \includegraphics. В простейшем варианте, чтобы включить в печатный документ рисунок, записанный в графическом файле *gr-file*, достаточно в нужное место входного файла вставить команду \includegraphics{*gr-file*}, указав в её аргументе имя файла.

```
---\includegraphics{a.eps}---
```



Однако решение реальной задачи не всегда сводится к столь простым рецептам. В частности, последний пример вызовет ошибку при использовании компилятора pdflatex:

```
? ! LaTeX Error: Unknown graphics extension: .eps.
See the LaTeX manual or LaTeX Companion for explanation. Type  H
<return>  for immediate help.
 ...
```

```
l.532 ---\includegraphics{a.eps}
                                   --- \label{a.eps}
?
```

Оказывается, что `pdflatex` не умеет импортировать рисунки в формате EPS (Encapsulated PostScript). Как нетрудно догадаться, для компилятора `pdflatex` более всего подходят рисунки в формате PDF. Но они непригодны для компилятора `latex`. Если документ PDF получать в два этапа, сначала выполнив компиляцию исходного текста в формат DVI (рис. 1.2), а затем преобразовать полученный `dvi`-файл в документ PDF с помощью программы `dvipdfm` (рис. 1.9), то проблема несовместимости графических форматов полностью устраняется.

Однако такое решение вряд ли можно признать идеальным хотя бы потому, что рисунки PDF занимают на диске компьютера значительно меньше места, чем рисунки EPS. Есть и другие причины для постепенного перехода к прямому компилированию документов LaTeX в формат PDF. Существуют простые рецепты, как подготовить рисунки и исходный текст с разметкой LaTeX так, чтобы сделать документ одинаково пригодным для обоих компиляторов. Поэтому советуем Читателю набраться немного терпения и вкратце познакомиться с тем, какие графические форматы существуют в настоящее время. Мы перечислим преимущественно те форматы, с которыми могут работать программы из библиотеки MiKTeX.

10.3.1. Форматы графических файлов

Все рисунки можно разделить на две категории: растровые и векторные. Растровое изображение состоит из матрицы пикселов, т. е. точек, упорядоченных по рядам и столбцам. Векторное изображение строится из отрезков линий и закрашенных областей; при выводе на экран монитора или принтер оно всё равно преобразуется в растровый вид, но в файле хранится в виде команд, описывающих форму и цвет геометрических объектов, что гарантирует сохранение качества изображения при изменении разрешения экрана или принтера.

Каждый пиксел всегда имеет один определённый цвет, а богатство цветовой палитры, то есть максимальное количество разных цветов, определяется глубиной цвета. Например, чёрно-белое изображение состоит из точек всего лишь двух цветов. Оно имеет глубину цвета 1 бит или, как говорят, закодировано в одной цветовой плоскости. Изображение глубиной 32 бита закодировано в 32-х цветовых плоскостях и может иметь до 2^{32} различных цветов.

Пример растровых рисунков даёт формат PCX, впервые реализованный в программе PC Paintbrush. Продолжая список наиболее распространённых растровых форматов, можно назвать BMP, GIF, JPEG, PNG, TGA и TIFF.

BMP Точечные рисунки BMP (Windows Bitmap, файлы с расширением `bmp`), используется в операционных системах Windows. Они могут содержать изображения с глубиной цвета 1, 2, 4, 8, 16 и 32 бита. Размер рисунка не ограничен.

GIF Формат GIF (Graphics Interchange Format, `gif`) предназначен для обмена данными через интернет и, соответственно, имеет высокую степень сжатия. Поддерживает до 256 цветов (2^8): изображение может быть чёрно-белым, 16-цветным, серым глубиной 8 бит, иметь цветовую палитру 8 бит. Отдельные цвета могут быть объявлены прозрачными. Позволяет сохранять несколько изображений в одном файле. Когда изображения быстро сменяют друг друга, GIF называется анимированным.

JPEG Формат JPEG (Joint Photographic Experts Group, `jpg`) разработан как схема сжатия информации для компьютерной графики. JPEG поддерживает цвета глубиной до 32 бит и используется, главным образом, для записи фотографических и сканированных изображений. В зависимости от степени сжатия качество изображения может изменяться от очень низкого до очень высокого.

PCX Изображение в формате PCX (PaintBrush, `pcx`) может быть чёрно-белым, 16-цветным, серым глубиной 8 бит, иметь цветовую палитру 8 бит или RGB цвета глубиной 24 бита. Формат PCX считается устаревшим. Вместо него рекомендуется использовать рисунки PNG.

PNG Формат PNG (Portable Network Graphics, `png`). Предназначен для обмена данными через интернет и, соответственно, имеет высокую степень сжатия. Предназначен для замены форматов GIF и TIFF, по сравнению с которыми имеет улучшенные свойства цветопередачи на различных выходных устройствах. Поддерживает до 32 цветовых плоскостей. Выбранные цвета могут быть объявлены прозрачными.

TGA Формат TGA (Targa, `tga`) используется для хранения растрового изображения. Может иметь глубину цвета до 24 бит.

TIFF Изображения в формате TIFF (Tagged Image File Format, расширение `tif` или `tiff`) используется для сохранения и обмена графическими данными различными операционными системами и приложениями. Может иметь глубину цвета 1, 4, 8, 24, 48 бит. Использует различные алгоритмы сжатия.

Недостатком растровых форматов является потеря качества при масштабировании изображения. Например, при увеличении изображения в два раза по высоте и ширине каждая точка делится на 4 (2 точки по горизонтали и 2 точки по вертикали) того же цвета, если не производится сглаживание резких переходов цвета (anti-aliasing). При сильном увеличении изображение становится зернистым либо же кажется нерезким (при сглаживании цвета). При уменьшении изображения некоторые линии могут вообще исчезнуть.

Векторные изображения лишены подобных недостатков, так как сохраняются в виде геометрического описания объектов, составляющих рисунок. Эти изображения могут также включать фрагменты растровых рисунков. В векторных форматах число битовых цветовых плоскостей заранее не определено.

Примером векторной графики является язык описания страниц PostScript. Форматы PostScript и EPS, использующие язык PostScript, получили очень широкое распространение. Продолжая список векторных графических форматов, пригодных для импорта в документы LaTeX, назовем ещё WMF, PDF и MPS.

EPS Формат EPS (Encapsulated PostScript, `eps`) является усечённой (инкапсулированной) версией PostScript. Используется для создания иллюстраций в настольных издательских системах. Один `eps`-файл может содержать изображение только одной страницы, но на ней может быть размещено несколько рисунков. Файлы в форматах PS и EPS занимают много места на диске компьютера, однако ряд программ может работать с `ps`- и `eps`-файлами, которые сжаты программой `gzip`. Сжатые файлы могут иметь расширение соответственно `ps.gz` и `eps.gz`.

MPS Изображения в формате MPS (MetaPost, `mps`) создаются программой MetaPost. Формат MPS является «родным» для системы LaTeX, так как он базируется на системе генерации шрифтов METAFONT для LaTeX. Из-за ограниченности сферы применения формата MPS мы не рассматриваем его в нашей книге.

PDF Изображения в формате PDF (Portable Document Format, `pdf`) импортируются в документ LaTeX при его компиляции с помощью `pdflatex`. Легко могут быть получены из рисунков EPS с помощью программы `epstopdf`. По сравнению с `eps`-файлами имеют существенно меньший размер.

PS Универсальный язык описания печатных страниц PostScript (Interpreted PostScript, `ps`) широко применяется в профессиональной печати. Один `ps`-файл может содержать изображение любого числа страниц, однако в документ LaTeX можно импортировать только такие `ps`-файлы, которые содержат одну страницу. Часто рисунки, записанные в файлах с расширением `ps`, на самом деле являются рисунками EPS.

WMF Формат WMF (Windows Metafile, `wmf`) разработан Microsoft Corporation как внутренний формат для Microsoft Windows 3. Используется для хранения как векторной, так и растровой графики. Может иметь глубину цвета на 24 бит. В документах LaTeX'а полностью уступил место формату EPS, так что требуются определённые усилия, чтобы импортировать рисунок WMF даже при работе в операционной системе Windows.

Программа `pdflatex`, которая компилирует исходный текст в разметке LaTeX в документ формата PDF, может импортировать рисунки в форматах PDF, PNG и JPEG[3].

Набор форматов, пригодных для компилятора `latex`, не столь однозначен, поскольку `latex` создаёт на выходе документ в формате DVI, который содержит

[3] Поддержка формата TIFF исключена начиная с `pdflatex` версии 1.10b.

не сам рисунок, а всего лишь ссылку на файл с рисунком. Теоретически `latex` может «импортировать» любой рисунок — важно, чтобы DVI-обозреватель мог отображать рисунки соответствующего формата.

На практике реализуется схема с конвертацией рисунков «на лету». Например, DVI-обозреватель `YAP` из библиотеки MiKTeX всегда показывает рисунки формата BMP, конвертируя при необходимости в этот формат рисунки других типов. Чтобы эта схема действовала, необходимо иметь подходящий набор графических фильтров, т. е. программ конвертации. Например, при установке программы Ghostscript `YAP` без дополнительных настроек приобретает умение показывать рисунки EPS, а если позаимствовать программу `giftopnm` из библиотеки программ netpbm, то удаётся импортировать рисунки GIF.

В составе MiKTeX имеются фильтры для рисунков PCX, PNG, TIFF и TGA. Эти и многие другие фильтры имеются в библиотеки netpbm, которая распространяется свободно.

В разделе 10.6 мы покажем, как импортировать рисунки «нестандартных» форматов.

10.3.2. EPS, или «В тесных рамках BoundingBox»

При вставке рисунка в печатный документ, как правило, необходимо задать размер рисунка. Проще всего подобрать размер для рисунка, записанного в форматах EPS или PDF. Дело в том, что полный размер рисунков этих форматов записывается в текстовом виде непосредственно в графический файл. Поэтому компилятор документов LaTeX способен прочитать этот размер прямо из файла с рисунком. При работе с другими графическими файлами пользователь, как правило, должен явно указать размер изображения в необязательном аргументе команды `\includegraphics`.

В заголовке `eps`-файла содержится строка вида

`%%BoundingBox:` *llx lly urx ury*

которая описывает ограничивающий бокс (BoundingBox), т. е. естественный (до масштабирования) размер рисунка и его положение на воображаемом листе бумаги. Вот что содержит файл `a.eps`, который был импортирован в виде рисунка на стр. 236:

```
%!
%%BoundingBox:100 100 136 136
100 100 moveto        % перейти в точку (100,100)
36 36 rlineto         % провести диагональную линию
36 neg 0 rlineto      % провести горизонтальную линию
36 36 neg rlineto     % провести другую диагональную линию
stroke                % нарисовать эти линии
100 100 moveto        % перейти в точку (100,100)
/Times-Roman findfont % загрузить шрифт Times-Roman
36 scalefont          % выбрать размер шрифта
```

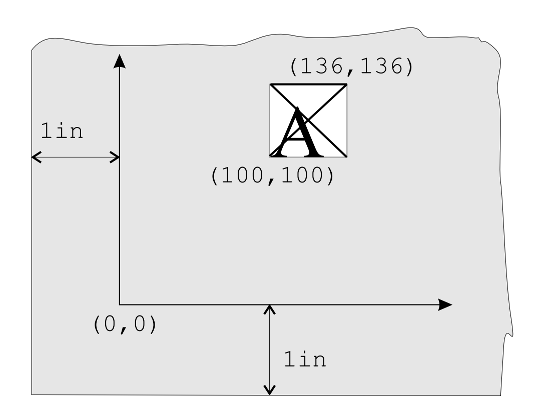


Рис. 10.2. Внутренняя система координат графического файла в формате EPS. Начало координат находится на расстоянии 1 дюйм от правого и нижнего краёв воображаемого листа бумаги. Изображение сосредоточено на белом фоне ограничивающего бокса

```
setfont              % сделать его текущим шрифтом
(A) show             % нарисовать букву A
showpage             % закончить страницу
```

Четыре числа во второй строке этого файла задают соответственно ординату, абсциссу левого нижнего угла и ординату, абсциссу правого верхнего угла бокса, внутри которого целиком помещаются все составные части изображения, записанного в файле. Координаты в `eps`-файле измеряются в «больших пунктах» bp (раздел 2.10) и отсчитываются от начала координат, которое привязано к воображаемому листу бумаги так, как показано на рис. 10.2. В файле `a.eps` ограничивающий бокс имеет левый нижний угол с координатами $(100, 100)$ и верхний правый угол с координатами $(136, 136)$. Таким образом, рисунок имеет ширину и высоту в $136 - 100 = 36$ bp, или 1,27 см. Команда `\includegraphics{a.eps}` выделяет для размещения рисунка, записанного в `a.eps`, прямоугольник с размерами 36×36 bp, причём точка (100,100) служит точкой привязки.

Файл формата PostScript в отличие от EPS, вообще говоря, содержит изображение сразу нескольких страниц, тогда как LaTeX может импортировать по одному рисунку за раз. Программа GSview позволяет преобразовывать `ps`-файлы в набор `eps`-файлов. Если известно, что `ps`-файл содержит только один рисунок, то преобразование в EPS формат в простых случаях можно выполнить, что называется, вручную, просто добавив строку с BoundingBox в заголовок файла. Часто файлы с расширением `ps` на самом деле являются рисунками в формате EPS.

Иногда один `eps`-файл может содержать два изображения: для воспроизведения с обычным и с высоким разрешением. В этом случае расположение второго изображения описывается в строке, начинающейся с `%%HiResBoundingBox`. Выбор нужного изображения производится ключом `hiresbb`, который описан в разделе 10.3.4.

10.3.3. Делаем PDF

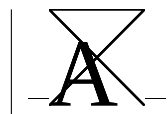
Почти все графические редакторы умеют сохранять рисунки в формате EPS, но ни одна известная нам графическая программа не позволяет сделать то же самое в формате PDF. Однако совсем нетрудно конвертировать рисунок EPS в формат PDF при помощи программы `epstopdf`. Например,

```
epstopdf a.eps
```

из `a.eps` сделает рисунок `a.pdf` в формате PDF. Программа `epstopdf` использует исполняемые модули программы Ghostscript, которая распространяется бесплатно, но отдельно от системы LaTeX. Заметим, что использование коммерческой программы Adobe Distiller для подобной конвертации нецелесообразно, так как Adobe Distiller создаёт `pdf`-файл с размером страницы, который не соответствует размерам рисунка, указанным в строке с BoundingBox. Кстати, BoundingBox в `pdf`-файле имеет другое название, а именно MediaBox, однако для дальнейшего это отличие не существенно, поскольку оно автоматически учитывается драйвером `pdftex`.

Чтобы вставить конвертированный рисунок `a.pdf`, нужно в примере на странице 236 в аргументе команды `\includegraphics` вместо `a.eps` указать `a.pdf`. После такой замены документ можно компилировать с помощью компилятора `pdflatex`. Однако перед компиляцией с помощью `latex` тогда придётся вновь редактировать исходный текст, чтобы сменить расширение имени импортируемого файла. К счастью, есть универсальный способ, как сделать исходный текст одинаково пригодным для компиляции любым компилятором. Нужно удалить расширение имени импортируемого файла вместе с точкой!

```
---\includegraphics{a}---
```

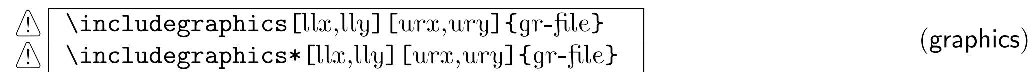


Компилятор сам выберет наиболее подходящее расширение. Правила, которыми он при этом руководствуется, описаны в разделе 10.6.

Далее везде, где не указано расширение имени импортируемого графического файла, предполагается, что выбор подходящего формата предоставлен компилятору.

10.3.4. Импорт векторных рисунков

Если рисунок записан в файле *gr-file*, то его можно вставить в печатный документ с помощью одной из команд



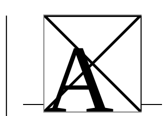
```
\includegraphics[llx,lly][urx,ury]{gr-file}
\includegraphics*[llx,lly][urx,ury]{gr-file}
```

(graphics)

Расширение имени вставляемого файла *gr-file* указывать необязательно. Далее мы так и будем делать в большинстве случаев, предполагая, что любой графический файл имеется в двух форматах, EPS и PDF. Первый из них будет выбран

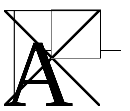
компилятором `latex`, второй — компилятором `pdflatex`. Если необязательный аргумент в команде `\includegraphics` пропущен, рисунок будет вставлен с его естественными размерами, т. е. размерами ограничивающего бокса. Чтобы пояснить это, повторим пример со стр. 236, обозначив ограничивающий бокс рамкой:

```
---\includegraphics{a}---
```



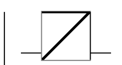
Опции [*llx*,*lly*], [*urx*,*ury*] позволяют изменить размер бокса, который LaTeX выделяет для размещения рисунка. Если присутствуют обе опции, то они задают соответственно нижний левый угол и верхний правый угол ограничивающего бокса в системе координат `eps`-файла в единицах длины, принятых в LaTeX'е (раздел 2.10). По умолчанию, когда единицы измерения длины не указаны, таковыми считаются большие пункты `bp`[4]. Поэтому `[1in,1in]` эквивалентно `[72,72]`. Если указана только одна опция, то она задаёт координаты верхнего правого угла [*urx*,*ury*], а координаты нижнего левого угла принимаются за нуль. Точкой привязки рисунка служит нижний левый угол ограничивающего бокса. Посмотрим, что произойдёт, если нижний левый угол ограничивающего бокса в предыдущем примере сдвинуть на четверть дюйма вверх и вправо:

```
---\includegraphics[118,118][136,136]{a}---
```



Место, выделенное LaTeX'ом под рисунок, для наглядности мы вновь обвели рамкой, поэтому хорошо видно, что рисунок «наехал» на окружающие его тире. Это произошло потому, что заданный нами ограничивающий бокс оказался меньше рисунка. Лишнюю часть можно обрезать, использовав `*`-форму команды `\includegraphics`:

```
---\includegraphics*[118,118][136,136]{a}---
```



Если Читатель попробует испытать приведённые примеры с вырезанием части рисунка, он обнаружит, что `latex` и `pdflatex` дают разные результаты. Дело в том, что при преобразовании рисунка EPS в рисунок PDF изменяются координаты рисунка в системе координат графического файла. Так, если в файле `a.eps` изображение занимало квадрат с координатами углов (100,100), (136,136), как показано на рис. 10.2, то в файле `a.pdf`, полученном с помощью программы `epstopdf` из `a.eps`, то же изображение будет помещено в квадрат с углами (0,0), (36,36). Результат не изменился бы, если исходное изображение имело эти коор-

[4] Такое же «правило по умолчанию» действует для любых других команд из графических пакетов коллекции graphics. В частности, можно не указывать размерность длины в аргументах команд `\rotatebox` и `\resizebox`, которые обсуждались в разделах 10.1 и 10.2.3. Однако это правило на самом деле является исключением, поскольку не действует для команд базового формата LaTeX.

динаты, то есть было бы прижато к левому нижнему углу воображаемой страницы. Ещё одно различие проявляется при использовании `\includegraphics` с двумя необязательными аргументами. В то время, когда мы писали эту главу, при выборе драйвера `pdftex` в последнем примере в рамку попадал нижний левый квадрант рисунка (что, по-видимому, является ошибкой), тогда как при выборе драйвера `dvips` — верхний правый. Далее мы покажем, что вырезать часть изображения можно с помощью ключей `trim` или `viewport`, причём этот способ всегда даёт одинаковый результат для рисунков как EPS, так и PDF.

При вставке графического изображение в формате, отличном от EPS или PDF, использование аргументов [*llx*,*lly*] и [*urx*,*ury*] фактически становится обязательным[5], так как иначе компилятор не может получить информацию о размере изображения. Однако не будем отвлекаться и продолжим разговор о рисунках на примере формата EPS, отложив обсуждение других форматов и других драйверов до раздела 10.3.6.

Расширенный графический пакет graphicx вводит более гибкий синтаксис команды `\includegraphics`:

```
⚠ \includegraphics[keyval-list]{gr-file}
⚠ \includegraphics*[keyval-list]{gr-file}
```
(graphicx)

Обязательный аргумент *gr-file* должен содержать название импортируемого файла. Необязательный аргумент *keyval-list* может содержать список ключей, перечисленных через запятую. Сами ключи записываются в виде равенств, в левой части которых стоит параметр, а в правой — его значение. Одного и того же результата можно добиться, используя разный набор ключей.

Начнём с ключей, которые задают размер ограничивающего бокса:

`bb` — параметр, используемый для задания всех координат углов ограничивающего бокса. Значение параметра должно содержать четыре числа, разделённых пробелами; запись `bb=`*a b c d* эквивалентна набору ключей `bbllx=`*a*, `bblly=`*b*, `bburx=`*c*, `bbury=`*d* (см. ниже). Например, `bb=0 0 1in 2in` определяет ограничивающий бокс с координатами нижнего левого угла (0,0) и с координатами верхнего правого угла — (72,144). Если единица измерения длины в координатах не указана, она считается равной `bp`;

`bbllx`, `bblly` — абсцисса и ордината нижнего левого угла ограничивающего бокса;

`bburx`, `bbury` — абсцисса и ордината верхнего правого угла ограничивающего бокса;

`natwidth`, `natheight` — естественные ширина и высота рисунка; ключи `natwidth` и `natheight` удобны для установки координат верхнего правого угла в случае, когда обе координаты нижнего левого угла равны нулю. Набор ключей `natwidth=`*w*, `natheight=`*h* эквивалентен `bb=`0 0 *w h*;

[5] Правило определения размеров рисунков может быть также задано глобально для всех рисунков сразу (раздел 10.5).

`hiresbb` указывает, что LaTeX должен считывать размеры ограничивающего бокса из строки, начинающейся с `%%HiResBoundingBox`, где обычно записывается второе изображение для воспроизведения на устройстве с высоким разрешением, а не из строки с `%%BoundingBox`. Ключ `hiresbb=true` эквивалентен простому заданию параметра `hiresbb` без значения. В случае, когда по умолчанию считываются размеры изображения с высоким разрешением (раздел 10.9.2), ключ `hiresbb=false` позволяет импортировать первый рисунок.

Так как `\includegraphics` автоматически считывает размеры ограничивающего бокса из `eps`-файла, перечисленные выше ключи обычно не используются при импортировании рисунков формата EPS или PDF, если отсутствует необходимость в выделении части рисунка, записанного в файле.

Следующая группа ключей применяется для обрезания ненужных частей рисунка:

`clip` может принимать значения `true` или `false`. При `clip=true` часть рисунка, выходящая за границы видимой области (см. ниже), отсекается. Если параметр `clip` присутствует в списке ключей, но не определён, то предполагается, что `clip=true`;

`viewport` задаёт видимую часть рисунка. Подобно параметру `bb`, видимая часть рисунка определяется двумя парами координат нижнего левого и верхнего правого углов видимой области. Координаты углов видимой области отсчитываются от левого нижнего угла ограничивающего бокса. Например, `viewport= 0 0 72 72` объявляет видимой областью квадрат высотой и шириной 1 дюйм, примыкающий к нижнему левому углу воображаемой страницы (т. е. к началу координат графического файла);

`trim` даёт альтернативный метод для выделения видимой части рисунка. Четыре числа задают смещение границ видимой области последовательно от левой, нижней, правой и верхней границ ограничивающего бокса. Положительные числа соответствуют смещению границы видимой области внутрь ограничивающего бокса, отрицательные числа расширяют видимую область. Например, `trim=1mm 2mm 3mm 4mm` отрезает от рисунка 1 мм слева, 2 мм снизу, 3 мм справа и 4 мм сверху;

`draft` устанавливает черновой режим импортирования, когда вместо рисунка строится рамка с размерами ограничивающего бокса, а внутри неё печатается название импортируемого файла. Ключ `draft=true` эквивалентен простому перечислению ключа `draft` без параметра. В случае, когда черновой режим используется по умолчанию (разделы 10.5 и 10.9.2), ключ `draft=false`, напротив, позволяет реально импортировать отдельный рисунок.

Хотя ключ `bb` в сочетании с `clip` может использоваться для обрезания лишних частей рисунка, рекомендуется применять `viewport` или `trim`. Мы уже видели, что неаккуратное переопределение размеров ограничивающего бокса приводит к выходу рисунка за его пределы, что обычно нежелательно. Ключ `clip` делает

фактически ненужным существование *-формы команды \includegraphics, но она сохранена для совместимости со стандартной версией графического пакета graphics (просто к списку ключей команды \includegraphics* автоматически добавляется clip). Для ещё большей совместимости команда \includegraphics с двумя необязательными аргументами автоматически использует синтаксис, вводимый пакетом graphics, даже если загружена расширенная версия пакета graphicx.

Для изменения масштаба и вращения рисунка используется другая группа ключей:

scale — изменение размера рисунка по заданному масштабу. Ключ scale=2 увеличивает рисунок в два раза относительно его естественного размера (заданного в строке с BoundingBox в eps-файле или ключом bb). Пропорции рисунка сохраняются;

width — требуемая ширина рисунка;

height | totalheight — требуемая высота или полная высота рисунка; задание любого из параметров width, height, totalheight приводит к масштабированию рисунка по заданному размеру;

keepaspectratio — масштабирование рисунка с сохранением пропорций. Рисунок увеличивается настолько, насколько это возможно при сохранении его аспектного (высота/ширина) отношения без выхода изображения за заданные размеры видимой области. Ключ keepaspectratio=true эквивалентен заданию ключа keepaspectratio без значения. Аспектное отношение сохраняется также и том случае, когда заданы только требуемая ширина или только требуемая высота;

angle — угол поворота рисунка (в градусах); положительное значение задаёт вращение против часовой стрелки;

origin определяет положение оси вращения рисунка; по умолчанию рисунок вращается относительно точки привязки. Возможные значения параметра origin показаны на рис. 10.1 на стр. 234. Например, origin=c устанавливает ось вращения в центр рисунка.

Наконец, последняя группа ключей относится к случаю, когда расширение имени импортируемого графического файла не указано:

type — тип рисунка. Все рисунки одного типа обрабатываются драйвером одинаковым образом. Фактически тип — это набор инструкций для преобразования ключей команды \includegraphics в команды программы-драйвера, которая собственно и печатает изображение на выходном устройстве. Обычно драйвер содержит набор инструкций для нескольких типов графических файлов. Например, при выборе драйвера dvips файлы с расширениями eps, ps, pz, eps.Z, ps.Z, eps.gz, ps.gz рассматриваются как векторные рисунки типа EPS, а файлы с расширениями bmp, mps и pcx считаются принадлежащими к типу растровой графики BMP. Поэтому следует задать ключ type=bmp

при попытке вставить рисунок `gif`. Драйвер `pdftex` различает графику трёх типов: PDF (файлы с расширениями `pdf`), MPS (`mps`) и PNG (`png`);

`ext` определяет расширение имени графического файла, если имя файла *gr-file* в обязательном аргументе указано без расширения. Например, если задан ключ `ext=.eps`, то драйвер выходного устройства попытается загрузить файл с именем *gr-file*`.eps`. Как правило, используется в совокупности с ключом `type`. Некоторые драйверы, в том числе `dvips`, игнорирует ключ `ext`, если не известно значение `type`;

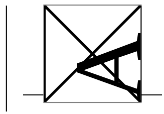
`read` определяет расширение имени файла, в котором записаны размеры ограничивающего бокса в виде строки `%%BoundingBox:` *llx lly urx ury* и который считывается LaTeX'ом при компиляции исходного текста печатного документа. Для графического файла с расширением `eps` размеры считываются из этого же файла, поэтому применение ключа `read=.eps` повторяет правило, используемое по умолчанию;

`command` — команда, которую программа-драйвер должна применить к графическому файлу перед его импортированием. Обычно используется для распаковки «на лету» сжатых (запакованных) рисунков EPS. Например, для файлов с расширениями `pz`, `eps.Z`, `ps.Z`, `eps.gz`, `ps.gz` драйвер `dvips` определяет правило, которое эквивалентно команде «`‘gunzip -c #1`».

Перейдём к примерам. Повернём сначала уже известный нам рисунок на 90°, предполагая, что загружен пакет graphicx:



```
---\includegraphics[angle=90]{a}---
```

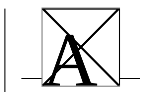


При загрузке стандартного графического пакета graphics тот же результат достигается более трудоёмким способом:

```
---\rotatebox{90}{\includegraphics{a}}---
```

который также проходит и с расширенным графическим пакетом graphicx. Теперь изменим размер рисунка, сохранив аспектное отношение:

```
---\includegraphics[width=0.4in]{a}---
```

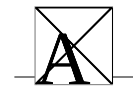


Тот же результат получается посредством

```
\resizebox{0.4in}{!}{\includegraphics{a}}
```

Покажем, как вставить рисунок, не превысив заданных размеров и сохранив аспектное отношение:

```
---\includegraphics[width=0.4in,height=4in,
        keepaspectratio]{a.eps}---
```



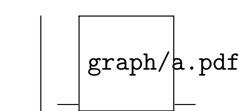
Высота рисунка не достигла заказанной величины в 4 дюйма, так как иначе ширина рисунка превысила бы заказанное значение в 0,4 дюйма или были бы искажены пропорции рисунка. Повторить этот пример при использовании средств стандартного пакета `graphics` предоставляем Читателю. Действие ключа `trim` в сочетании с `clip` иллюстрирует следующий пример:

```
---\includegraphics[trim=-2 -4 6 8,clip]{a}---
```



Наконец, продемонстрируем действие ключа `draft`:

```
---\includegraphics[draft,width=0.5in]{a}---
```



Тот же результат получается, если графический пакет загружен с опцией `draft`, только эта опция действует на все команды `\includegraphics`. Ключ `draft=false` позволяет реально вставить рисунок даже в этом случае.

Примеры с использованием ряда других ключей имеются в разделе 10.3.6, где обсуждается вставка растровых рисунков.

10.3.5. Импорт запакованных рисунков

Размер рисунков EPS (или PostScript), как правило, бывает неоправданно большим. При сжатии размер файлов часто удаётся уменьшить раз в 10. Многие реализации LaTeX'а могут производить распаковку сжатых рисунков что называется «на лету». Можно воспользоваться такой возможностью и постоянно хранить рисунки в сжатом виде.

Покажем, как это сделать, на примере графического файла с именем `tiger.ps`, который распространяется вместе с программой Ghostscript (рисунок на стр. 268). Найдите в файле `tiger.ps` строку, содержащую размер рисунка. Для этого откройте файл `tiger.ps` в любом текстовом редакторе (например, в блокноте notepad) и найдите строку, начинающуюся с `%%BoundingBox`. Она располагается в первых строках файла. Скопируйте эту строку во вновь созданный файл с именем `tiger.ps.bb`. Должно получиться нечто вроде

```
%%BoundingBox: 0 0 567 738
```

Разумеется, числа, которые собственно задают размеры, индивидуальны для каждого рисунка. Теперь можно запаковать файл `tiger.ps` программой `gzip`:

```
gzip tiger.ps
```

Он создаст запакованный рисунок `tiger.ps.gz` и удалит `tiger.ps`. Обратную операцию распаковки выполняет программа `gunzip`.

Чтобы вставить запакованный рисунок в документ LaTeX, достаточно указать имя запакованного файла в аргументе команды `\includegraphics`, причём лучше всего опустить расширение имени:

```
\includegraphics{tiger}
```

Как мы объяснили в разделе 10.3.3, компилятор определит, графический файл с каким расширением следует использовать. В частности, компилятор `latex` в том случае, когда явно или неявно при загрузке пакета graphics (или graphicx) выбран драйвер `dvips`, сначала попытается найти файл `tiger` с расширением `.ps`. Не найдя этот файл, компилятор попытается найти файл `tiger.ps`, но он был удален программой `gzip`. Затем компилятор обнаружит `tiger.ps.gz`. Тогда он прочитает размер запакованного рисунка из файла `tiger.ps.bb`, зарезервирует в `dvi`-файле свободное место для размещения рисунка и запишет команду, которую DVI-обозреватель должен будет выполнить, чтобы показать рисунок на экране монитора или распечатать на принтере. Эта команда включает ссылку на программу `gunzip`, которая должна находиться в известном DVI-обозревателю месте (например, её можно поместить в тот каталог, где размещены все исполняемые программы системы LaTeX).

Размеры запакованного рисунка можно также указать в необязательном аргументе команды `includegraphics` в виде значений параметров `bb` или `natwidth` и `natheight`. Этот способ продемонстрирован в следующем разделе на примере растровых рисунков.

10.3.6. Импорт растровых рисунков

Чтобы в печатный документ вставить растровый рисунок, его размеры необходимо указать в необязательном аргументе команды `\includegraphics` или же в специальном текстовом файле с расширением `bb`, аналогично тому, как это было показано выше на примере вставки запакованных рисунков EPS.

Предположим, что загружен расширенный графический пакет graphicx. Надеемся, что Читатель сумеет переформулировать приводимые ниже примеры на язык стандартного графического пакета graphics.

Если ставить своей целью создание документа, одинаково пригодного для компиляторов `latex` и `pdflatex`, то среди множества форматов растровой графики лучше всего выбрать рисунки PNG. Все современные редакторы растровых рисунков умеют экспортировать изображение в формат PNG. Это современный формат растровой графики, он совмещает в себе достоинства более старых форматов, но лишён многих их недостатков. Рисунки PNG имеют малый размер и широко используются при оформлении веб-страничек в интернете.

Вставим изображение льва. Пусть оно находится в файле `lion.png` и имеет полный размер 125 на 404 пиксела. Поскольку высота рисунка великовата, приведём сразу достаточно сложный пример, показав, как можно выделить только небольшую часть рисунка. Исходные размеры рисунка (его BoundingBox) зада-

дим с помощью ключей **natwidth** и **natheight**, а видимую часть — с помощью ключа viewport:

```
\includegraphics[bb=0 0 125 404,
    viewport=0 224 125 404, clip=true,
    scale=0.5]{lion.png}
```



Ключ clip здесь добавлен для того, чтобы часть рисунка вне размеров viewport стала невидимой. После всех урезаний размер рисунка ещё уменьшен в два раза с помощью ключа scale. Вместо ключа bb с равным успехом можно было бы использовать ключи natwidth и natheight, лишнее убрать при помощи ключа trim:

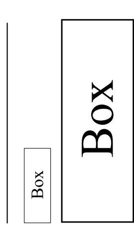
```
\includegraphics[natwidth=125, natheight=404,
    trim=0 224 0 0, clip=true, scale=0.5]{lion.png}
```

Мы указали полное имя файла (вместе с расширением), поскольку драйвер dvips (в отличие от pdftex) в своём списке расширений по умолчанию не имеет этого расширения. В разделе 10.6 мы покажем, как восполнить этот пробел.

10.4. Вращение плюс масштабирование

Ключи в необязательном аргументе \includegraphics читаются слева направо. Если не учитывать это обстоятельство, можно получить неожиданный результат. В следующем примере левый бокс поворачивается на 90° и затем масштабируется до высоты 1 см.

```
\includegraphics[angle=90,totalheight=1cm]{box}
\includegraphics[totalheight=1cm,angle=90]{box}
```



Правый бокс сначала увеличивается до той же высоты, а затем поворачивается.

Нужно также быть внимательным при использовании ключа height. Ошибки возникают, когда под высотой графического объекта height подразумевают полную высоту, которая на самом деле устанавливается параметром totalheight, а не height. Конечно, если объект имеет нулевую глубину depth, как у левого бокса на рис. 10.3, то totalheight=height, и проблем не возникает. Если объект имеет ненулевую глубину (средний бокс на рис. 10.3), а вместо totalheight при

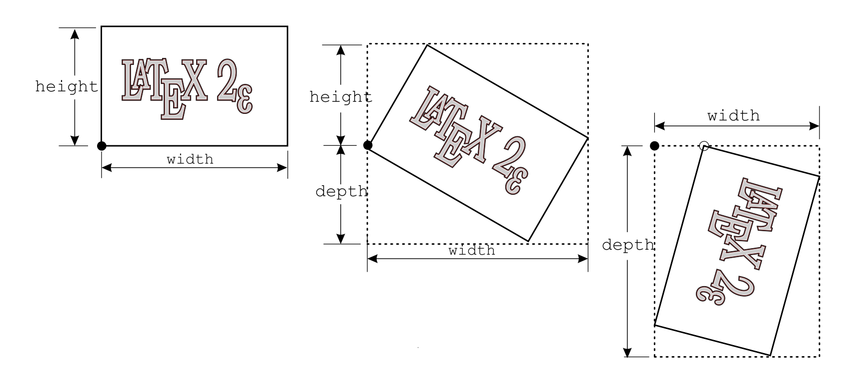


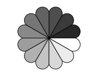
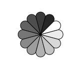
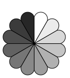
Рис. 10.3. Определение высоты повёрнутого рисунка. Точка привязки изображена кружком, а точка вращения (в случае несовпадения с точкой привязки) — окружностью

масштабировании рисунка использован ключ `height`, то рисунок станет слишком большим. При нулевой высоте `height` (правый бокс на рис. 10.3) масштабирование рисунка приводит к ошибке деления на нуль, так как LaTeX вычисляет масштабирующий коэффициент, производя деление заказанной высоты рисунка на величину `height`.

10.4.1. От перестановки слагаемых сумма меняется

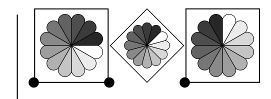
Параметры `totalheight`, `height` и `width` задают размеры ограничивающего бокса, а не самого рисунка, находящегося в боксе. Это особенно важно понимать, осуществляя поворот и одновременное масштабирование рисунка. Например, следующие три рисунка розеточки во входном файле заказаны с одинаковыми размерами:

```
\includegraphics[totalheight=1cm]{rosette}
\includegraphics[angle=45,totalheight=1cm]{rosette}
\includegraphics[angle=90,totalheight=1cm]{rosette}
```

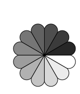
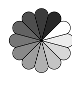
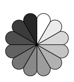
Парадокс легко разрешается, если взглянуть на то, что происходит с ограничивающими боксами рисунков:

```
\includegraphics[totalheight=1cm]{rosette}
\includegraphics[angle=45,totalheight=1cm]{rosette}
\includegraphics[angle=90,totalheight=1cm]{rosette}
```



Каждый рисунок здесь масштабируется так, что его ограничивающий бокс после поворота имеет полную высоту 1 см. Чтобы получить три розеточки одного размера, надо переставить порядок ключей в необязательном аргументе `\includegraphics`:

```
\includegraphics[totalheight=1cm]{rosette}
\includegraphics[totalheight=1cm,angle=45]{rosette}
\includegraphics[totalheight=1cm,angle=90]{rosette}
```

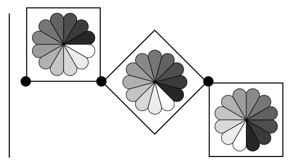
  

Теперь с размерами всё в порядке, но рисунки расположились на разной высоте. О выравнивании рисунков по вертикали мы поговорим в следующем разделе.

10.4.2. Выравнивание рисунков по вертикали

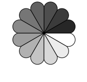
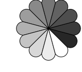
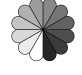
Попробуем в последнем примере изменить направление вращения на противоположное, заменив для угла поворота `angle` значения 45 и 90 соответственно на -45 и -90.

```
\includegraphics[totalheight=1cm]{rosette}
\includegraphics[totalheight=1cm,angle=-45]{rosette}
\includegraphics[totalheight=1cm,angle=-90]{rosette}
```



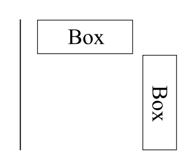
Проблемы с выравниванием рисунков по высоте только усугубились. Чтобы понять причину происшедшего, вновь следует посмотреть на то, что происходит с боксами при заказанном преобразовании рисунков. Мы предусмотрительно изобразили вместе с рисунками их ограничивающие боксы и точки привязки. Точки привязки (первоначально ими были нижние левые углы) располагаются на одной — базисной — линии. В этом-то и всё дело! Если требуется, чтобы выравнивались центры рисунков, наиболее простым решением является использование ключа `origin=c` с вращением импортируемого рисунка относительно его центра:

```
\includegraphics[totalheight=1cm]{rosette}
\includegraphics[totalheight=1cm,origin=c,angle=-45]
  {rosette}
\includegraphics[totalheight=1cm,origin=c,angle=-90]
  {rosette}
```

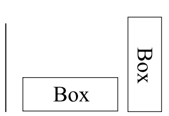
Правильный выбор оси вращения чаще всего решает все проблемы с размещением рисунков. Например, при вращении рисунка по часовой стрелке на 90° он располагается ниже базисной линии:

```
\includegraphics[width=0.5in]{box}
\includegraphics[width=0.5in,angle=-90]{box}
```



Чтобы рисунок оказался выше базисной линии, достаточно сделать поворот вокруг нижнего правого угла:

```
\includegraphics[width=0.5in]{box}
\includegraphics[width=0.5in,origin=br,angle=-90]{box}
```



10.5. Глобальная установка ключей

Значения ключей в необязательном аргументе *keyval-list* команды `\rotatebox` или `\includegraphics` (в варианте расширенного графического пакета graphicx) могут быть установлены при помощи декларации

`\setkeys{`*operaion*`}{`*keyval-list*`}` (keyval, graphicx)

Она определена в пакете keyval, который автоматически загружается пакетом graphicx и служит для глобальной установки списка ключей *key=value*, используемых по умолчанию командами `\rotatebox` и `\includegraphics`. Список ключей в *keyval-list* разделяется запятыми, так же как в необязательном аргументе этих команд. Если *operation* имеет значение `Grot`, список ключей предназначен команде `\rotatebox`; `Gin` адресует ключи команде `\includegraphics`.

Например, чтобы угол поворота в команде `\rotatebox` по умолчанию измерялся в радианах, достаточно ввести

```
\setkeys{Grot}{units=6.28318}
```

Аналогично, если требуется изменить масштаб всех рисунков так, чтобы они занимали ровно 75% ширины страницы, достаточно ввести

```
\setkeys{Gin}{width=0.75\textwidth}
```

Область действия `\setkeys` определяется по стандартным правилам группирования. Поэтому последующие команды `\includegraphics` внутри текущей группы или процедуры будут действовать так, как будто ключ `width=0.75\textwidth` стоит самым первым в списке ключей *keyval-list* этой команды. Другие ключи, явно указанные в `\includegraphics`, просто добавляются к нему. Следующий пример показывает, как ввести черновой режим импортирования последующих рисунков:

```
\setkeys{Gin}{draft=true}
```

Эта строка, будучи помещённой в преамбулу входного файла, устанавливает черновой режим для всех импортируемых рисунков. Тот же эффект достигается, если загрузить пакет graphicx с опцией `draft`. Обсуждение других опций графических пакетов мы отложим до раздела 10.9.

10.6. Операции с графическими файлами

LaTeX ищет графические файлы там же, где он ищет входной файл. Чтобы расширить область поиска, достаточно использовать декларацию

`\graphicspath{`*dir-list*`}` (graphics, graphicx)

перечислив в её аргументе *dir-list* дополнительные каталоги, причём каждый должен быть заключён в фигурные скобки. Например, декларация

```
\graphicspath{{eps/}{png/}}
```

заставляет LaTeX при поиске импортируемых графических файлов дополнительно просматривать подкаталоги `eps` и `png` в каталоге, который является текущим (обычно это каталог, где находится входной файл).

Декларация

`\DeclareGraphicsExtensions{`*exts*`}` (graphics, graphicx)

определяет действия системы в случае, когда в команде `\includegraphics` имя файла *gr-file* указано без расширения. Аргумент *exts* должен содержать список расширений имени файла, перечисленных через запятую (причём пробелы игнорируются, а точка в расширении обязательна). Полное имя импортируемого файла получается добавлением к *gr-file* первого расширения из списка *exts*. Если файл найден, на этом поиск заканчивается, а если не найден, делается попытка найти файл со следующим расширением из списка *exts* и т. д.

При выборе драйвера `pdftex` по умолчанию устанавливается порядок поиска файлов, который эквивалентен

```
\DeclareGraphicsExtensions{.png,.pdf,.jpg,.mps}
```

Встретив команду `\includegraphics{`*gr-file*`}`, компилятор `pdflatex` пробует сначала загрузить файл *gr-file*`.png`, затем *gr-file*`.pdf`, затем *gr-file*`.jpg`, затем *gr-file*`.mps`. Драйвер `dvips` по умолчанию использует другое правило

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z}
```

и в первую очередь пытается найти файл с расширением `eps`.

Если печатный документ содержит много рисунков, то отказ от явного указания расширения имени импортируемого файла открывает путь к глобальному управлению импортированием, поскольку можно регулировать не только очерёдность поиска файлов по их расширению, но и правила обработки файлов по их расширениям. Например, если опустить расширение имён файлов с рисунками EPS, то рисунки в любой момент можно сжать, упаковав с помощью программы `gzip`. При этом исходный текст не придётся изменять, чтобы импортировать запакованные файлы (см. раздел 10.3.5).

Если расширение имени файла не указано в команде `\includegraphics`, то графический файл должен существовать в момент обработки входного файла, иначе невозможно определить, какое расширение из списка *exts* нужно выбрать. Напротив, если расширение явно указано, то импортируемый файл может отсутствовать в момент обработки входного файла (например, в это время он может находиться в запакованном виде). Это замечание не относится к компилятору `pdflatex`, поскольку в отличие от `latex` он внедряет рисунки в выходной файл, поэтому они должны быть в наличии на момент компиляции. Компилятор должен также иметь возможность определить размер рисунка, и это ещё одна причина, по которой импортируемый файл должен существовать в момент компиляции

входного файла, если информация о размерах изображения считывается прямо из него.

Декларация

`\DeclareGraphicsRule{`*ext*`}{`*type*`}{`*read*`}{`*command*`}` (graphics, graphicx)

определяет, как `\includegraphics` обрабатывает импортируемый файл в зависимости от его расширения *ext*, причём неважно, указано расширение явно в команде `\includegraphics` или взято из списка расширений по умолчанию. Можно использовать несколько деклараций `\DeclareGraphicsRule`, по одной для каждого расширения. Аргументы *ext*, *type*, *read* и *command* почти аналогичны одноимённым ключам в команде `\includegraphics` (раздел 10.3.4).

ext — начинающееся с обязательной точки расширение имени файла. Как особый случай, в качестве *ext* можно задать `*`, чтобы ввести правило импортирования файлов с любым недекларированным расширением (см. пример ниже).

type — тип рисунка для этого расширения. Все файлы одного типа импортируются посредством одних и тех же внутренних команд, определённых драйвером. Например, файлы с расширениями `ps`, `eps`, `pz` все классифицируются как файлы типа `eps`.

read — начинающееся с обязательной точки расширение имени файла, который содержит информацию о размере ограничивающего бокса для рисунка, записанного в файле с расширением *ext*. Значение *read* может совпадать с *ext*, но может и отличаться. Например, файл с расширением `.ps.gz` (запакованный рисунок PostScript) не является текстовым и не может быть прочитан LaTeX'ом во время компиляции входного файла. Поэтому информацию о размере ограничивающего бокса можно поместить в текстовый файл с расширением `.ps.bb`, указав это расширение в аргументе *read*. Если этот аргумент пуст, информация о размерах рисунка, импортируемого из файла с расширением *ext*, должна быть определена в необязательном аргументе команды `\includegraphics`.

`command` — команда, которая должна быть применена к файлу с расширением *ext* перед импортированием рисунка (часто отсутствует). Перед командой должна стоять обратная кавычка ` (не путать с более привычной прямой кавычкой '). В аргументе *command* параметр `#1` используется для обозначения имени импортируемого файла.

Например, следующее правило

```
\DeclareGraphicsRule{.pz}{eps}{.bb}{`gunzip -c #1}
```

определяет, что любой файл с расширением `pz` обрабатывается как сжатый графический файл типа EPS. Информация об ограничивающем боксе для файлов с расширением `pz` хранится в файлах с расширением `bb`, а команда `gunzip -c` «на

лету» распакует рисунок (так как компилятор `latex` не может читать информацию об ограничивающем боксе из сжатого файла, строка с BoundingBox должна быть в незапакованном файле).

Множество подобных правил составляет то, что называется драйвером графического пакета. Напомним, что имя драйвера указывается в необязательном аргументе команды `\usepackage`, котрая загружает любой графический пакет.

Драйверы определяют процедуру импортирования файлов каждого типа, поэтому прежде чем изобретать собственную процедуру, полезно ознакомиться с существующей. Для этого нужно просмотреть файл настройки драйвера. Это обычный текстовый файл с расширением `def`. Вот как можно было бы воспроизвести заданные драйвером `dvips` правила обработки файлов с расширениями `eps` и `ps`[6]:

```
\DeclareGraphicsRule{.eps}{eps}{.eps}{}
\DeclareGraphicsRule{.ps}{eps}{.ps}{}
```

Оба вида файлов отнесены к типу `eps`. Растровые рисунки, записанные в файлах с расширениями `pcx` и `bmp`, тем же драйвером классифицируются как рисунки типа `bmp`:

```
\DeclareGraphicsRule{.pcx}{bmp}{}{}
\DeclareGraphicsRule{.bmp}{bmp}{}{}
```

Поскольку третий аргумент здесь пуст, компилятор `latex` даже не пытается прочитать размеры рисунка из какого-либо файла — они должны быть указаны в самой команде `\includegraphics`.

Драйвер `dvips` использует ещё несколько правил подобного вида, из которых два заслуживают особого внимания в связи с разделом 10.3.5:

```
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{`gunzip -c #1}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
```

Они устанавливают, что размеры рисунка из запакованных файлов с расширениями `ps.gz` и `eps.gz` считываются из текстовых файлов с расширениями, соответственно, `ps.bb` и `eps.bb`, а для распаковки сжатых файлов используется программа `gzip`.

Файлы с расширениями, которые никак не описаны, отнесены драйвером `dvips` к типу `eps`. Это декларирует команда

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

Она же гласит, что компилятор должен попытаться извлечь информацию о размерах рисунка непосредственно из графического файла (напомним, что это обычно невозможно).

[6] В файле драйвера `dvips.def` эти правила записаны более сложным образом через команды «низшего уровня».

Драйвер `pdftex`, который автоматически выбирается компилятором `pdflatex`, действует по иным правилам, причём они ещё и зависят от версии формата PDF выходного файла, куда записывается откомпилированный документ. Каждый графический формат из числа поддерживаемых компилятором `pdflatex` отнесён к отдельному типу:

```
\DeclareGraphicsRule{.png}{png}{.png}{}
\DeclareGraphicsRule{.pdf}{pdf}{.pdf}{}
\DeclareGraphicsRule{.jpg}{jpg}{.jpg}{}
\DeclareGraphicsRule{.mps}{mps}{.mps}{}
```

Теперь можно переходить к выводам и «усовершенствованиям».

Во-первых, полезно заметить, что рисунки PNG, которым драйвер `pdftex` отвёл первое место, драйвер `dvips` относит к разряду «прочие», поскольку они вообще не упоминаются в `dvips.def`. Однако исполняемые программы из библиотеки MiKTeX «на лету» конвертируют рисунки PNG в рисунки BMP, с которыми этот драйвер знает что делать. Поэтому среди растровых рисунков использование формата PNG наиболее предпочтительно с точки зрения совместимости компиляторов `latex` и `pdflatex`.

Во-вторых, с помощью декларации `\DeclareGraphicsRule` можно добавить новые правила, хотя обычно это затрудняет поочерёдное использование компиляторов `latex` и `pdflatex` без переделки исходного текста. Для примера покажем, как добавить правило для конвертирования «на лету» рисунка из формата GIF, который иначе невозможно импортировать в документ LaTeX:

```
\DeclareGraphicsRule{.gif}{bmp}{}{}
```

Это правило подходит для драйвера `dvips`, но не для `pdftex`, поскольку последний не умеет обращаться с рисунками BMP. При установке дополнительных графических фильтров можно импортировать рисунок практически любого формата. Например, если имеется библиотека ImageMagick, можно импортировать в документ PDF рисунок формата TIFF, поддержка которого исключена из новых версий компилятора `pdflatex`:

```
\DeclareGraphicsRule{.tif}{png}{.png}{‘convert #1 ‘basename #1 .tif‘.png}
```

Однако на практике проще конвертировать рисунок в подходящий формат заранее с помощью одного из графических редакторов.

Чтобы разместить большой рисунок, импортированный при помощи команды `\includegraphics`, не оставляя заметных пустых промежутков на странице, приходится перемещать рисунок относительно окружающего текста. LaTeX позволяет автоматизировать процесс подбора подходящего места для рисунка. Как это сделать, рассказывает следующая глава.

10.7. Пакет color

Загрузим пакет color

```
\usepackage{color}
```

предполагая, что компилятор `latex` выберет драйвер `dvips`, а `pdflatex` — драйвер `pdftex`.

Надеемся, что Читатель уже освоил технику работы с графикой. Если он сумел воспроизвести на своём оборудовании примеры из первой части этой главы, то, скорее всего, ему не придётся удивляться, почему на цветном мониторе печатный документ остается чёрно-белым.

10.7.1. Цветовые модели

Существует несколько технических систем записи цвета, которые иногда называются цветовыми моделями. LaTeX поддерживает четыре цветовые модели.

`rgb` (Red-Green-Blue — красный-зелёный-синий) — модель, в которой цвет идентифицируется тремя десятичными числами от 0 до 1, задающими интенсивность красного, зелёного и синего *базовых* цветов. Например, чисто красный цвет записывается как (1,0,0). Белый цвет (1,1,1) состоит из всех трёх базовых цветов максимальной интенсивности. Чёрный цвет (0,0,0) получается при выключении всех базовых цветов.

Все другие цвета также получаются смешиванием трёх базовых цветов, причём красному цвету соответствует излучение с длиной волны 700 нанометров (1 нм = 10^{-9} м), зелёному — 546,1 нм, а синему — 435,8 нм. Цветовая модель `rgb` применяется в телевизорах и цветных мониторах компьютеров, где поток электронов, вылетающих с катода электронно-лучевой пушки, ударяет по люминофору экрана, а зёрна люминофора высвечивают излучение с заданной длиной волны.

`cmyk` (Cyan-Magenta-Yellow-blacK — голубой-пурпурный-жёлтый-чёрный) — модель, в которой цвет идентифицируется четырьмя числами от 0 до 1, задающими интенсивность базовых цветов в соответствии с аддитивной моделью, используемой в большинстве принтеров. Получила широкое распространение в цветной полиграфии.

`gray` — модель серой шкалы. Одно десятичное число от 0 (чёрный) до 1 (белый) задаёт оттенки серого цвета. Модель `gray` используется при печати полутоновых чёрно-белых изображений.

`named` — цветовая модель, в которой используемым цветам присваиваются имена, например JungleGreen. Технически не реализована ни в одном устройстве. Под именем цвета просто скрывается набор чисел, соответствующий реально используемой цветовой модели `rgb`, `cmyk` или `gray`. Соответствие имени

реальному цвету устанавливается драйвером. Драйвер `dvips` содержит определения 68 наименований цветов[7]. Этот набор названий можно сделать доступным при использовании других драйверов, если пакет color загрузить с опцией `dvipsnames`.

Цветовая модель `named` является не более чем одним из примеров цветовых моделей, которые распознают цвета не по набору чисел, а по именам. В локальных реализациях LaTeX'а могут быть введены и другие цветовые модели, такие как Pantone (промышленный стандарт цветов) или X11 (названия цветов в системе XWindows). Во входном файле можно также ввести наименования цветов при помощи декларации `\definecolor`, которая описана в разделе 10.7.2.

Далеко не все драйверы поддерживают весь набор цветовых моделей. Если какая-то цветовая модель не поддерживается драйвером, при обработке входного файла LaTeX выдаёт сообщение об ошибке. Например, следующее сообщение

```
! LaTeX Error: Undefined color model ‘rgb’
```
[8]

буквально гласит, что не определена цветовая модель `rgb`, но фактически означает, что драйвер вообще не поддерживает работу с цветом.

10.7.2. Назовите цвет

Цвета `black` (чёрный), `white` (белый), `red` (красный), `green` (зелёный), `blue` (синий), `cyan` (голубой), `magenta` (пурпурный), `yellow` (жёлтый) должны быть предопределены пакетом color. Если необходимо расширить рабочую палитру цветов, можно ввести дополнительные названия цветов с помощью декларации

⚠ `\definecolor{`*name*`}{`*model*`}{`*clr*`}` (color)

где *name* есть название цвета, которое можно в дальнейшем использовать в командах переключения цвета, *model* — цветовая модель (`rgb`, `cmyk`, `gray` или `named`), а *clr* — соответствующая ей спецификация цвета, т. е. набор чисел или ранее заданное название цвета. Например, следующие декларации

```
\definecolor{MyBlue}{rgb}{0.8,0.85,1}
\definecolor{MyOrange}{cmyk}{0,0.42,1,0}
\definecolor{MyGray}{gray}{0.75}
\definecolor{MyGreen}{named}{OliveGreen}
```

[7] GreenYellow, Yellow, Goldenrod, Dandelion, Apricot, Peach, Melon, YellowOrange, Orange, BurntOrange, Bittersweet, RedOrange, Mahogany, Maroon, BrickRed, Red, OrangeRed, RubineRed, WildStrawberry, Salmon, CarnationPink, Magenta, VioletRed, Rhodamine, Mulberry, RedViolet, Fuchsia, Lavender, Thistle, Orchid, DarkOrchid, Purple, Plum, Violet, RoyalPurple, BlueViolet, Periwinkle, CadetBlue, CornflowerBlue, MidnightBlue, NavyBlue, RoyalBlue, Blue, Cerulean, Cyan, ProcessBlue, SkyBlue, Turquoise, TealBlue, Aquamarine, BlueGreen, Emerald, JungleGreen, SeaGreen, Green, ForestGreen, PineGreen, LimeGreen, YellowGreen, SpringGreen, OliveGreen, RawSienna, Sepia, Brown, Tan, Gray, Black, White.

[8] Ошибка LaTeX'а: Неопределённая цветовая модель «rgb».

вводят четыре рабочих цвета, после чего MyBlue, MyOrange, MyGray, MyGreen можно использовать в командах переключения цвета, причём цвет MyGreen будет идентичен OliveGreen из набора dvipsnames (чтобы последняя из четырёх деклараций сработала, пакет color должен быть загружен с опцией dvips или dvipsnames). В следующем разделе мы увидим, что употребление двух названий MyGreen и OliveGreen одного и того же цвета может различаться в зависимости от способа загрузки пакета color.

10.7.3. Раскрашивание текста

Синтаксис команд переключения цвета текста подобен синтаксису команд переключения шрифта и имеет две формы: декларативную и командную.

```
\color[model]{clr}
\textcolor[model]{clr}{text}
```
(color)

Область действия декларации \color определяется обычными правилами группирования. Иными словами, она ограничивается ближайшей парой фигурных скобок, внутри которой стоит \color.

```
На {\color{MyGreen} зелёном} лужке
\textcolor{blue}{синие} цветочки.
```

На зелёном лужке синие цветочки.

Команда \textcolor[model]{clr}{text} эквивалентна более длинной конструкции {\color[model]{clr} text}.

На тот случай, если Читатель не увидит на экране своего монитора синих цветочков на зелёном лужке, попытавшись повторить последний пример, приведём полный текст входного файла:

```
\documentclass{article}
\usepackage[dvipsnames]{color}
\definecolor{MyGreen}{named}{OliveGreen}
\begin{document}
   На {\color{MyGreen} зелёном} лужке
   \textcolor{blue}{синие} цветочки.
\end{document}
```

Необязательный аргумент model в командах \color и \textcolor не нужен, если цвет clr был предварительно описан при помощи \definecolor. В нашем примере цвет MyGreen явно определён во входном файле, а цвет blue задан пакетом color. При явном указании на цветовую модель cmyk тот же результат достигается следующим образом:

```
На {\color[cmyk]{0.64,0,0.95,0.40}зелёном} лужке
\textcolor[cmyk]{1,1,0,0}{синие} цветочки.
```

На зелёном лужке синие цветочки.

Чтобы использовать поименованный цвет из набора 68 цветов dvipsnames, следует явно указать, что используется цветовая модель named:

`На {\color[named]{OliveGreen} зелёном} лужке \textcolor[named]{Blue}{синие} цветочки.`	На зелёном лужке синие цветочки.

Поскольку предопределённый пакетом color цвет `blue` (не путать с `Blue`!), как и любой цвет, объявленный при помощи `\definecolor`, не принадлежит к модели `named`, команда

```
\textcolor[named]{blue}{синие}
```

привела бы к ошибке:

```
! Package color Error: Undefined color ‘blue’⁹.
```

Опцию `[named]` в `\textcolor[named]{Blue}{...}` можно опустить, если пакет color загружен с опцией `usenames`; то же самое относится ко всем другим командам, которые воспринимают цвет.

10.7.4. Цветные боксы

⚠ `\colorbox[`*model*`]{`*clr*`}{`*lr-text*`}`
⚠ `\fcolorbox[`*model*`]{`*fclr*`}{`*clr*`}{`*lr-text*`}` (color)

Команда `\colorbox` печатает аргумент *lr-text* в строковой моде в боксе, у которого цвет фона задан аргументами *model* и *clr*.

`На \colorbox{green}{зелёном} лужке \colorbox[named]{Cyan}{синие} цветочки.`	На зелёном лужке синие цветочки.

Команда `\fcolorbox` дополнительно обводит бокс рамкой цвета *fclr*. Если указана цветовая модель *model*, она относится к обоим аргументам *clr* и *fclr*.

`На \fcolorbox{red}{green}{зелёном} лужке \fcolorbox[named]{Red}{Cyan}{синие} цветочки.`	На зелёном лужке синие цветочки.

Команда `\fcolorbox` использует параметры `\fboxrule` и `\fboxsep`, чтобы определить толщину рамки и размер бокса так же, как это делает команда `\fbox` (разделы 9.1 и 9.1.4).

10.7.5. Цветные страницы

Цвет фона текущей страницы и всех последующих изменяет декларация

⚠ `\pagecolor[`*model*`]{`*clr*`}` (color)

Её область действия глобальна, то есть не ограничивается фигурными скобками. Чтобы вернуть белый цвет фону страницы, нужно использовать команду `\pagecolor{white}`.

[9] Ошибка пакета color: Неопределённый цвет «blue».

10.8. Другие пакеты в коллекции graphics

В коллекцию графических пакетов входят ещё два пакета, написанные на основе graphicx.

10.8.1. Пакет epsfig

Пакет epsfig оставлен для совместимости со старыми версиями печатных документов. Для импортирования PostScript-рисунков он вводит команду

```
\epsfig{file=gr-file, height=h, width=w, angle=angle,
       bbllx=bblx, bblly=bbly, bburx=bbrx, bbury=bbry,
       clip=, silent=, rheight=rh, rwidth=rw}
```
(epsfig)

где только первый ключ `file=`*gr-file* является обязательным (вместо `file` можно писать `figure`). Другие ключи, за исключением `silent`, `rheight` и `rwidth`, уже описаны в разделе 10.3.4. Ключ `silent` переводит команду `\epsfig` в режим работы, препятствующий выводу служебной информации в файл протокола. Ключи `rheight` и `rwidth` задают высоту и ширину бокса, который LaTeX должен зарезервировать под рисунок (по умолчанию значения `rheight` и `rwidth` совпадают с размерами рисунка `height` и `width`). Знак `=` у ключей `clip` и `silent` обязателен, хотя им не нужно присваивать какие-либо значения. Порядок расположения ключей в аргументе команды `\epsfig` не имеет значения.

Декларация

```
\psfigdriver{driver}
```
(epsfig)

определяет драйвер, который будет использоваться для перевода `dvi`-файла в `ps`-файл. По умолчанию используется драйвер, указанный в файле настройки `graphics.cfg` или в опции декларации `\usepackage` при загрузке пакета. Декларации

```
\psdraft        \psfull
\psscalefirst   \psrotatefirst
```
(epsfig)

регулируют работу команды `\epsfig`. Декларация `\psdraft` действует, как ключ `draft` в команде `\includegraphics`. Она устанавливает, что вместо рисунка нужно нарисовать рамку. Такой режим отменяется другой декларацией `\psfull`, которая действует по умолчанию. Декларация `\psscalefirst` устанавливает, что сначала проводится масштабирование рисунка, а потом поворот. По умолчанию действует противоположная декларация `\psrotatefirst`.

10.8.2. Пакет lscape

Пакет lscape вводит процедуру

```
\begin{landscape} ... \end{landscape}
```
(lscape)

внутри которой тело страницы поворачивается на 90° и печатается в альбомной ориентации. Колонтитулы страницы и подстрочные примечания печатаются на обычном месте, как при портретной ориентации страницы.

10.9. Опции графических пакетов

Пакеты из коллекции graphics могут загружаться с опциями, которые модифицируют их функционирование. Часть опций является общей для всей коллекции, другая — специфична для отдельных пакетов. К первой группе принадлежат опции выбора драйвера выходного устройства.

10.9.1. Драйверы

Графические пакеты должны знать, какое устройство будет использоваться для печати документа или его просмотра на экране монитора. Список драйверов приведен в табл. 10.1. Имена драйверов указываются в необязательном аргументе декларации `\usepackage`. Следующий пример показывает, как загрузить пакет graphics для работы с драйвером `dvips`:

```
\usepackage[dvips]{graphics}
```

По умолчанию загружается драйвер, указанный в файле `graphics.cfg`, точно так же, как пакет color по умолчанию загружается с драйвером, заданным в файле настройки `color.cfg`.

Загружать пакеты более чем с одним драйвером бессмысленно, так как будет использован последний. Если несколько пакетов используют общий драйвер, его можно указать в необязательном аргументе декларации `\documentclass` (раздел 3.1).

10.9.2. Опции пакетов graphics и graphicx

Ряд опций можно использовать с пакетами graphics, graphicx, epsfig и lscape:

`draft` | `final` позволяют изменить действие аналогичных опций при выборе класса печатного документа в `\documentclass`. Если для обработки или просмотра печатного документа, содержащего множество рисунков, требуется немало времени, можно отменить реальное импортирование графических изображений, загрузив пакеты с опцией `draft`. В этом случае графические файлы используются компилятором только для получения информации о размерах импортируемого изображения, а в печатном документе вместо рисунка печатается имя графического файла в рамке с размерами, равными размеру рисунка. По умолчанию действует опция `final`, если она не замещена опцией `draft` в `\documentclass`;

`hiderotate` указывает, что не нужно показывать вращаемый графический объект (возможно потому, что драйвер не может вращать);

Таблица 10.1

Драйверы графических пакетов. Знаком × отмечены не поддерживаемые функции

Драйвер	Импорт графики	Вращение	Масштабирование	Цвет	Фирма / Автор
dvipdf	•	•	•	•	Лесенко, Сергей
dvips	•	•	•	•	Т. Рокицки (Rokicki, Tomas)
dvipsone	•	•	•	•	Y&Y, Inc.
dviwin	•	×	×	×	Х. Сендукас (Sendoukas, Hippocrates)
emtex	•	×	×	×	Э. Маттес (Mattes, Eberhard)
pctex32	•	•	•	•	Personal TEX, Inc.
pctexhp	•	×	×	×	Personal TEX, Inc.
pctexps	•	•	•	•	Personal TEX, Inc.
pctexwin	•	×	×	×	Personal TEX, Inc.
pdftex	•	•	•	•	Х. Хаген (Hagen, Hans)
psprint	•	×	×	×	Э. Треворов (Trevorrow, Andrew)
tcidvi	•	×	×	×	Scientific Word
textures	•	•	•	•	Blue Sky Research
truetex	•	×	×	•	Р. Кинч (Kinch, Richard)
vtex	•	•	•	•	MicroPress, Inc.

Примечание: вместо драйверов dviwindo, oztex, xdvi, имевшихся в ранних версиях коллекции пакетов graphics, следует использовать dvips.

hidescale указывает, что не нужно показывать масштабируемый графический объект (возможно потому, что драйвер не может масштабировать);

hiresbb указывает, что размеры ограничивающего бокса следует взять из строки, содержащей %%HiResBoundingBox, а не %%BoundingBox;

debugshow требуется при отладке пакета; обычному пользователю вряд ли придётся применять эту опцию.

10.9.3. Опции пакета color

Три из четырёх имеющихся опций пакета color контролируют работу с цветовой моделью named. Изначально эта модель не содержит заранее предопределённых названий цветов, типа тех 68, которые известны драйверу dvips. Опция dvipsnames заставляет пакет загрузить определения всех этих 68 поименованных цветов. Противоположная ей опция nodvipsnames предотвращает эту операцию. Опция usenames делает все загруженные поименованные цвета доступными непосредственно в командах переключения цвета, как мы объяснили в разделе 10.7.3.

Наконец, опция monochrome приводит к тому, что любые команды, которые описаны в разделе 10.7, будут проигнорированы. Эта опция необходима при использовании драйвера, который не поддерживает работу с цветом:

```
\usepackage[emtex,monochrome]{color}
```

Это может помочь при предварительном просмотре или распечатке пробных чёрно-белых копий.

Итак, помимо названия драйвера пакет color может иметь следующие опции:

dvipsnames | nodvipsnames управляет загрузкой 68 наименований цветов из набора dvipsnames, перечисленных на странице 259. По умолчанию эти цвета доступны (опция dvipsnames) при выборе драйвера dvips и не доступны (опция nodvipsnames) при выборе любого другого драйвера. Чтобы сделать исходный текст одинаково пригодным для компиляторов latex и pdflatex, следует всегда загружать пакет color с опцией dvipsnames;

usenames делает возможным использование названий цветов, загружаемых опцией dvipsnames, непосредственно в командах переключения цвета без явного указания цветовой модели named;

monochrome игнорирует команды переключения цвета.

Громада двинулась и рассекает волны.
Плывёт. Куда ж нам плыть?
А. Пушкин. Осень

Глава 11

Плавающие объекты

В отличие от обычного текста, рисунки и некоторые таблицы нельзя перенести на следующую страницу по частям. Подобные объекты называются боксами (глава 9). Чтобы избежать появления полупустых страниц, большие боксы должны уметь «уплывать» в подходящее место, например в верхнюю часть страницы, где они впервые упомянуты. Стандартные классы печатных документов располагают двумя процедурами `figure` и `table`, которые формируют плавающие объекты. Процедура `figure` обычно используется для размещения рисунков, созданных процедурой `picture` (раздел 9.4) или импортированных командой `\includegraphics` (раздел 10.3.4), а `table` — для таблиц, созданных процедурой `tabular` (глава 12). Однако, по большому счёту, компилятору LaTeX безразлично, для размещения каких объектов используются эти процедуры. Единственное различие между ними состоит в том, как они подписывают размещаемый объект: *Таблица*, *Рис.* или как-то иначе.

К плавающим объектам относятся также заметки на полях, которые создаются командой `\marginpar`. Поскольку размер заметок на полях обычно невелик, им не нужно далеко «уплывать», но они могут переходить с правого поля страницы на левое, в зависимости от чётности номера страницы.

11.1. Процедуры `figure` и `table`

Процедуры `figure` (рисунок) и `table` (таблица) располагаются в исходном тексте обычно там, где впервые упоминаются размещаемые ими объекты. LaTeX самостоятельно находит для них свободное место. Рисунок 11.1 в верхней части следующей страницы во входном файле описан следующим образом:

```
...место. Рисунок~\ref{fig:2}
\begin{figure}\center
  \includegraphics[height=5cm]{golfer}
  \caption{Подпись к рисунку} \label{fig:2}
\end{figure}
в верхней части следующей страницы во входном файле...
```

Тело процедуры `figure` в данном примере состоит из четырёх команд. Декларация `\center` (раздел 5.1) обеспечивает центрирование рисунка, который импор-



Рис. 11.1. Подпись к рисунку

тирует команда `\includegraphics`, причём её синтаксис предполагает, что загружен пакет graphicx (раздел 10.3.4). Подпись к рисунку создаёт команда `\caption`. Она же печатает порядковый номер рисунка. Команда `\label` (раздел 3.7), идущая вслед за `\caption`, метит рисунок, приписывая ему метку (в данном случае метку `fig:2`). По этой метке можно сослаться на рисунок в любом месте печатного документа (раздел 3.7).

LaTeX обрабатывает тело процедур `figure` и `table` в текстовой моде. Поэтому для размещения объектов внутри процедуры можно использовать процедуры типа `center`, команды `\hspace` и `\vspace`. Процедуры `figure` и `table` создают абзац такой же ширины, какую имеет окружающий их текст. В разделе 10.4 рассказано, как разместить два рисунка рядом друг с другом и ещё о многих других приёмах форматирования. В теле процедур `figure` и `table` может быть более одной команды `\caption`. Таким способом можно создавать единый плавающий объект с несколькими рисунками или таблицами, снабжёнными собственными подписями. Рис. 11.2 и 11.3 иллюстрируют эту возможность. Они содержат два тигриных портрета, которые во входном файле описаны следующим образом:

```
\begin{figure}
  \includegraphics[width=0.47\textwidth]{tiger}                    \hfill
  \reflectbox{\includegraphics[width=0.47\textwidth]{tiger}}       \\
  \parbox[t]{0.47\textwidth}{\caption{Это тигр Тай}\label{fig:3}} \hfill
  \parbox[t]{0.47\textwidth}{\caption{А это его брат-близнец Гер.
     \\ Он зеркальная копия Тая}\label{fig:4}}
\end{figure}
```

Ключи `width=0.47\textwidth` в аргументах `\includegraphics` указывают, что каждый портрет должен быть отмасштабирован так, чтобы его ширина составила 47% ширины страницы. Команда `\hfill` позиционирует портреты на равном расстоянии друг от друга. Каждый портрет снабжен своей подписью. Они имеют такую же ширину, как и сами рисунки. Изменение ширины подписей достигает-



Рис. 11.2. Это тигр Тай

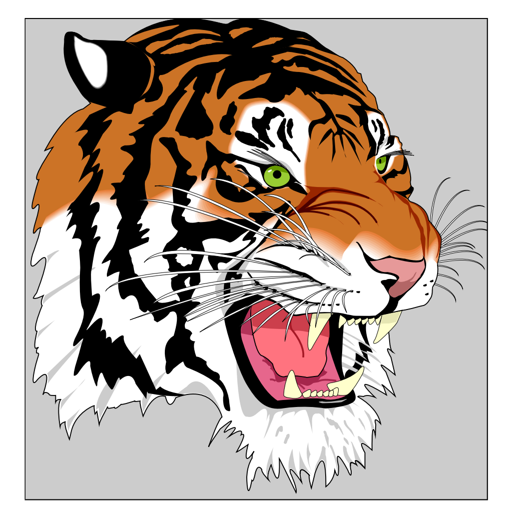


Рис. 11.3. А это его брат-близнец Гер. Он зеркальная копия Тая

ся за счёт размещения команды `\caption` в парбоксе `\parbox` заданной ширины. Опция `t` в команде `\parbox` обеспечивает выравнивание парбоксов по верхней строке; без неё подписи к рисункам начинаются на разной высоте, если содержат разное число строк. Наконец, обратим внимание Читателя, что точка в конце подписи к рисунку или заголовка таблицы не ставится, даже если они состоят из нескольких предложений. Издательства обычно требуют, чтобы заголовки таблиц размещались над ними, а подписи к рисункам — под рисунками. В первом случае команду `\caption` в теле соответствующей процедуры нужно поставить до описания самой таблицы после командной скобки `\begin{table}`, а во втором — после описания рисунка перед `\end{figure}`.

Команда `\caption` может использоваться только в теле процедур `figure` и `table`. Её аргумент с текстом подписи является подвижным, так как может быть включён в список рисунков или таблиц, который печатают соответственно команды `\listoffigures` и `\listoftables`. Поэтому хрупкие команды в аргументе команды `\caption` должны быть защищены командой `\protect`.

В печатном документе класса `article` (статья) или `proc` (доклад) используется сплошная нумерация рисунков и таблиц во всех разделах. Классы `book` (книга) и `report` (отчёт) устанавливают независимую нумерацию в пределах каждой главы. В классах `slides` (слайды) и `letter` (письмо) процедуры `figure` и `table` не существуют (они там и не нужны). Текущий номер рисунков и таблиц хранится в счётчиках, имена которых совпадают с именами процедур `figure` и `table` соответственно. Общие приёмы работы со счётчиками были описаны в разделе 2.9.

Приведём теперь полный синтаксис процедур `figure` и `table`.

`\begin{figure}[`*loc*`]`	*body*	`\end{figure}`
`\begin{figure*}[`*loc*`]`	*body*	`\end{figure*}`
`\begin{table}[`*loc*`]`	*body*	`\end{table}`
`\begin{table*}[`*loc*`]`	*body*	`\end{table*}`

При печати в две колонки процедуры `figure` и `table` выделяют место для бокса в одной колонке вне зависимости от его ширины, а `*`-формы процедур отводят место сразу в двух колонках. Обе формы эквивалентны в одноколоночном формате. Необязательный аргумент *loc* уточняет способ размещения плавающего объекта. Он может состоять из последовательности следующих четырёх спецификаторов:

`h` — здесь: разрешает размещение плавающего объекта после заполнения текущей строки (нельзя использовать в `*`-формах процедур при печати в две колонки);

`p` — плавающая страница: разрешает размещение на отдельной странице, содержащей только плавающие объекты;

`t` — вверху: выше текста на текстовой странице;

`b` — внизу: ниже текста на текстовой странице (нельзя использовать в `*`-формах процедур при двухколоночной печати).

Число спецификаторов в *loc* может изменяться от нуля до четырёх. Наличие всех спецификаторов (`htbp`) разрешает все варианты размещения, причём имеет значение порядок следования спецификаторов. Если необязательный аргумент пуст (`[]`) или вообще отсутствует, действует правило «по умолчанию». Стандартные классы устанавливают, что в этом случае используется последовательность `tbp`, так что рисунок или таблица будут размещены в верхней части текстовой страницы, нижней её части или на отдельной плавающей странице, содержащей только подписи к рисункам и таблицам.

Опция *loc* может также содержать восклицательный знак `!`, отменяющий некоторые ограничения на размещение плавающих объектов (см. ниже). Поскольку этот спецификатор не имеет самостоятельного значения, его можно вставлять в *loc* только при наличии хотя бы одного из спецификаторов `h`, `p`, `t` и `b`.

Если LaTeX размещает плавающие объекты в нежелательном месте, можно либо передвинуть их в другое место входного файла, либо явно наложить запрет на размещение дополнительных плавающих объектов на текущей странице при помощи декларации

`\suppressfloats[`*no-loc*`]`

Она действует от места расположения во входном файле до конца текущей страницы в печатном документе. Неразмещённые рисунки и таблицы передвигаются на следующую страницу. Если вставить `\suppressfloats`, например, после каждой процедуры `figure`, то на всех страницах (кроме плавающих) будет располагаться не более одного рисунка. Опция *no-loc* может содержать либо `t` (запрещает размещение дополнительных рисунков и таблиц в верхней части текстовой

страницы), либо `b` (запрещает размещение дополнительных рисунков и таблиц в нижней части текстовой страницы). Если опция *no-loc* отсутствует, то запрет на размещение плавающих объектов распространяется на верхнюю и нижнюю части страницы, однако знак `!` в аргументе *loc* процедур `figure` и `table` отменяет для конкретного рисунка или таблицы запрет на размещение, наложенный декларацией `\suppressfloats`.

Как это всё сложно — может подумать наш Читатель. Механизм размещения плавающих объектов в LaTeX'е действительно не прост, но он гарантирует отсутствие полупустых страниц, которые то и дело появляются в печатных документах, изготовленных другими текстовыми процессорами. LaTeX размещает плавающий объект на ближайшем месте, выбор которого не нарушает перечисленные ниже правила.

- Объект не может быть напечатан на странице, предшествующей тому месту в тексте, где находится соответствующая ему процедура `table` или `figure`.
- Рисунки и таблицы печатаются в той последовательности, в которой они описаны в тексте входного файла. Рисунок не может быть напечатан ранее предыдущего рисунка, а таблица — ранее предыдущей таблицы. Однако при печати в две колонки одноколоночный объект (процедуры `figure` и `table`) может быть размещён ранее двухколоночного (процедуры `figure*` и `table*`).
- Объект может быть напечатан только в положении, разрешённом спецификаторами в необязательном аргументе *loc*, или (если тот пропущен) спецификаторами `tbp`, принятыми по умолчанию, причём спецификатор `h` (здесь) имеет приоритет над `t` (вверху).
- Размещение рисунка не может привести к переполнению страницы.
- Границы страницы, определяемые описанными ниже параметрами, не нарушаются. Однако, если в опции *loc* имеется спецификатор `!`, ограничения, установленные для текстовых (но не плавающих) страниц, игнорируются.

Команды `\clearpage`, `\cleardoublepage` (раздел 4.7) и `\end{document}` принудительно печатают все необработанные рисунки и таблицы, отменяя действие последних трёх правил. Команда `\chapter`, определённая в стандартных классах `book` и `report`, где она открывает новую главу, автоматически исполняет команду `\clearpage` или `\cleardoublepage`, поэтому плавающие рисунки и таблицы ни при каких условиях не переходят в следующую главу.

При использовании необязательного аргумента *loc* следует включить в него достаточное количество опций, чтобы разрешить печать рисунка или таблицы хотя бы где-нибудь; иначе эти и следующие за ними рисунки и таблицы будут занимать память компьютера до конца главы или всего текста, возможно, вызывая её нехватку.

LaTeX форматирует тело процедуры размещения плавающих объектов, обозначенное выше как *body*, в виде парбокса (раздел 9.2). Его ширина равна ширине рабочего поля страницы или колонки окружающего текста, которая хранится в команде `\textwidth` (раздел 17.2), а высота определяется содержанием тела процедуры. В теле процедуры не могут находиться другие процедуры размещения плавающих объектов.

Команда

⚠ `\caption[`*entry*`]{`*head*`}`

печатает пронумерованную подпись к рисунку или таблице. Она имеет два аргумента:

entry — необязательный аргумент. Его содержимое вносится в список рисунков и таблиц, которые печатаются соответственно командами `\listoffigures` и `\listoftables` (раздел 3.10), т. е. это подвижный аргумент. По этой причине текст записи *entry* не должен содержать более пары сотен символов. Его также нельзя делить на абзацы; следовательно, текст не должен содержать пустых строчек. Если этот аргумент пропущен (вместе с квадратными скобками), вместо него в запись помещается текст *head*;

head — текст подписи. Если пропущен необязательный аргумент, текст из *head* переписывается в список рисунков или таблиц, будучи в этом случае подвижным аргументом. Если *head* содержит очень длинный текст или состоит из нескольких абзацев, необходимо использовать более короткий *entry*.

Команда `\caption` должна использоваться в текстовой моде, но может также помещаться в парбокс, создаваемый командой `\parbox` или процедурой `minipage` (глава 9).

Команда `\label{`*key*`}` (раздел 3.7), устанавливающая ключ *key* для ссылки на номер подписи, должна располагаться в *head* или после команды `\caption` в теле процедуры. По умолчанию алгоритм размещения плавающих объектов отдаёт предпочтение варианту с расположением их в верхней части страницы, даже если при этом объект появляется до фактической ссылки на него, т. е. до команды `\ref{`*key*`}` с тем же ключом *key*. Если это нежелательно, достаточно загрузить пакет flafter. Он гарантирует, что плавающий объект не будет напечатан до первой ссылки на него.

Параметры настройки

Изменение значений нижеследующих параметров настройки, сделанное в преамбуле, действует с первой страницы текста. Изменение, сделанное после `\begin{document}`, начинает действовать со следующей, а не с текущей страницы. Параметры, применяемые ко всем плавающим объектам при одноколоночной печати, применимы к одноколоночным плавающим объектам в двухколоночном формате.

topnumber — счётчик; максимальное число плавающих объектов, которые могут быть размещены в верхней части текстовой страницы.

dbltopnumber — аналог **topnumber** для двухколоночных плавающих объектов при двухколоночной печати.

bottomnumber — счётчик; максимальное число плавающих объектов, которые могут быть размещены в нижней части текстовой страницы.

totalnumber — счётчик; максимальное число плавающих объектов, которые могут быть размещены на одной странице вне зависимости от положения.

\topfraction — максимальная доля страницы, которая может быть занята плавающими объектами в её верхней части. Изменяется при помощи команды **\renewcommand**. Например, значение 0,25 устанавливает, что под размещение плавающих объектов может быть отведено до четверти страницы в её верхней части.

\dbltopfraction — аналог **\topfraction** для плавающих объектов, занимающих обе колонки на двухколоночной странице.

\bottomfraction — максимальная доля страницы, которая может быть занята плавающими объектами в её нижней части. Изменяется командой **\renewcommand**. Например, значение 0,33 устанавливает, что под размещение плавающих объектов может быть отведено до трети страницы в её нижней части.

\textfraction — минимальная доля страницы, которая должна быть занята текстом. Остаток (1 − **\textfraction**) могут занять плавающие объекты. Изменяется при помощи команды **\renewcommand**.

\floatpagefraction — минимальная доля плавающей страницы, подлежащая заполнению плавающими объектами; ограничивает размер пустого пространства на плавающей странице. Изменяется командой **\renewcommand**.

\dblfloatpagefraction — аналог **\floatpagefraction** для плавающих объектов, занимающих 2 колонки.

\floatsep — величина вертикального пробела, вставляемого между плавающими объектами в верхней или нижней части текстовой страницы. Растяжимая длина.

\dblfloatsep — аналог **\floatsep** для двухколоночных плавающих объектов на двухколоночной странице. Растяжимая длина.

\textfloatsep — величина вертикального пробела, вставляемого между плавающими объектами в верхней или нижней части страницы и текстом на этой странице. Растяжимая длина.

\dbltextfloatsep — аналог **\textfloatsep** для двухколоночных плавающих объектов на двухколоночной странице. Растяжимая длина.

\intextsep — величина вертикального пробела, вставляемого выше и ниже плавающего объекта, размещаемого в середине текстовой страницы в соответствии со спецификатором **h** в опции loc. Растяжимая длина.

\figurename — ключевое слово в подписи к рисунку. Изменяется с помощью **\renewcommand**.

\tablename — ключевое слово в подписи к таблице. Изменяется с помощью **\renewcommand**.

11.2. Подписи к рисункам и таблицам

Формат подписей к плавающим объектам определяют классы печатных документов, причём делают это очень жёстко: изменить этот формат, не изменив определение команды `\caption`, почти невозможно. Например, стандартные классы

после номера рисунка и таблицы ставят двоеточие, тогда как в литературе на русском языке принято ставить точку. Выход из такого рода затруднений открывает пакет caption и его более новая версия caption2. Оба пакета написаны Харальдом Зоммерфельдом (Sommerfeldt, Harald). Чтобы внести какое-нибудь изменение в формат подписей, достаточно всего лишь загрузить один из этих пакетов с соответствующей опцией. Например, при подготовке этой книги мы загружаем пакет caption2 с опциями `centerlast` и `small`:

```
\usepackage[centerlast,small]{caption2}
```

Первая из них приводит к тому, что последняя строчка в подписях центрируется, а наименование второй опции происходит от декларации переключения размера шрифта и означает, что подписи печатаются размером `\small`. Чтобы печатать подписи крупным шрифтом, следовало бы использовать опции `large` или `Large` соответственно декларациям `\large` и `\Large`. Чтобы подписи были напечатаны полужирным (`\bfseries`) или машинописным (`\ttfamily`) шрифтом, следует использовать опции `bf` или `tt`. Надеемся, что логика подбора опций Читателю понятна. Наименования других опций, достойных упоминания, совпадают с именами процедур `center`, `flushleft`, `flushright` (глава 5). Чтобы заменить злополучное двоеточие после номеров рисунков и таблиц в подписях, достаточно после загрузки пакета caption2 переопределить команду

`\captionlabeldelim` (caption2)

Например, после

```
\renewcommand{\captionlabeldelim}{.}
```

двоеточие будет заменено точкой.

Пакет caption2 имеет ещё много других возможностей для настройки формата подписей, однако он не входит в стандартный набор пакетов, распространяемых группой разработчиков формата LaTeX 2_ε, поэтому мы ограничимся только уже сделанными пояснениями. Нестандартных пакетов слишком много, чтобы мы имели возможность описать хотя бы их десятую часть. Тем не менее в следующем разделе мы упомянем ещё ряд таких пакетов[1].

11.3. Приёмы работы с плавающими объектами

Загрузив дополнительные пакеты, можно изменить способ размещения плавающих объектов с помощью процедур `figure` и `table` либо же добавить иные процедуры для размещения рисунков и таблиц.

[1] Метод установки дополнительных пакетов описан в разделе 3.3.7.

11.3.1. Пакет afterpage

В редких случаях, когда механизм позиционирования плавающих объектов бывает перегружен большим количеством рисунков и таблиц, ждущих своей очереди, приходится прибегать к помощи команды `\clearpage`. Мы уже упоминали, что она принудительно печатает все плавающие объекты, пренебрегая некоторыми ограничениями на их размещение. Недостаток такого метода заключается в том, что простое применение команды `\clearpage` может приводить к образованию полузаполненных страниц. Удобное решение даёт команда

`\afterpage{text}` (afterpage)

реализованная в пакете afterpage Дэвида Карлайла (Carlisle, David). Она исполняет команды, находящиеся в её аргументе *text* после завершения текущей страницы. Следовательно, `\afterpage{\clearpage}` принудительно печатает все плавающие объекты, в то время как текущая страница до конца будет заполнена текстом. Другой ценный пример применения команды `\afterpage` имеется в разделе 12.5. Пакет afterpage входит в число стандартных пакетов, т. е. имеется в любой минимальной реализации LaTeX'а.

11.3.2. Пакет endfloat

Редакции некоторых журналов требуют, чтобы в представленных к публикации статьях рисунки и таблицы были сгруппированы в конце документа. В подобных случаях рисункам и таблицам обычно должен предшествовать их список. Если редакция не подкрепляет свои требования специально разработанным классом документа, можно использовать пакет endfloat Джеймса Даррела (Darrel McCauley, James) и Джефа Голдберга (Goldberg, Jeff).

Он размещает рисунки и таблицы изолированно в конце статьи в отдельный раздел, озаглавленный соответственно Figures (Рисунки) и Tables (Таблицы). В начале этих разделов автоматически печатается список иллюстраций и таблиц. Печать списков можно отключить, поместив в преамбулу документа декларации `\nofiglist` и `\notablist` соответственно.

По месту исходного расположения процедур `figure` и `table` в текст документа вставляются примечания вида «[Figure 1 about here]». Эти примечания можно отключить, поместив в преамбулу декларацию `\nomarkersintext`. Текст примечания можно изменить, переопределив команды `\figureplace` и `\tableplace` при помощи `\renewcommand`. Например, сделать метки на русском языке можно так:

```
\renewcommand{\figureplace}{\figurename~\thefigure\ здесь}
\renewcommand{\tableplace}{\tablename~\thetable\ здесь}
```



а) Лай



б) Йон

Рис. 11.4. Семейка львов

11.3.3. Пакет subfigure

Пакет subfigure Стивена Кокрена (Cochran, Steven) упрощает размещение нескольких рисунков внутри одного плавающего объекта, причём помимо общей подписи каждый рисунок может иметь свою собственную подпись. Пакет вводит команду

`\subfigure[subhead]{subfigure}` (subfigure)

где обязательный аргумент *subfigure* предназначен для рисунка; если рисунок импортируется командой `\includegraphics`, то *subfigure* должен содержать эту команду. Если задан необязательный аргумент `[`*subhead*`]`, то рисунок будет иметь свой собственный номер, даже если `[`*subhead*`]` пуст, т. е. имеются только квадратные скобки `[]`. По умолчанию этот номер печатается в виде (а), но это правило можно изменить, переопределив команду `\thesubfigure`, которая печатает значение счётчика `subfigure`. Значение счётчика увеличивается каждой командой `\subfigure` независимо от наличия необязательного аргумента. Определение `\thesubfigure`, действующее по умолчанию, эквивалентно `(\alph{subfigure})`. С помощью команды `\subfigure` портреты ближайших родственников тигрят со страницы 268 можно вставить следующим образом:

```
\begin{figure}
\center
\renewcommand{\thesubfigure}{\asbuk{subfigure})}
\subfigure[Лай]{\includegraphics[width=0.47\textwidth]{lion2}}\hfill
\subfigure[Йон]{\reflectbox{\includegraphics[width=0.47\textwidth]{lion2}}}
\caption{Семейка львов}\label{fig:4a}
\end{figure}
```

Здесь команда `\thesubfigure` изменена так, чтобы нумерация внутренних рисунков более соответствовала принятой в отечественной литературе. Результат показан на рис. 11.4.

Размещение внутренних рисунков внутри плавающего объекта регулируется командами

```
\subfigtopskip
\subfigcapskip
```
(subfigure)

Первая из них хранит величину вертикального пробела, который вставляется сверху, между и снизу внутренних рисунков. Первоначально величина этого пробела равна `10pt`. Ещё один вертикальный пробел размером `\subfigcapskip` добавляется перед общей подписью.

11.4. Обтекание рисунков



Рис. 11.5. Мать Лая и Йона

Стандартные классы печатных документов не содержат средств размещения небольших рисунков и таблиц так, чтобы они обтекались окружающим их текстом (пример такого обтекания даёт рис. 11.5). Однако существует несколько пакетов, которые позволяют это делать. Они не входят в набор пакетов, поставляемых группой разработчиков формата LaTeX, поэтому мы только упомянем некоторые из них, не углубляясь в детали. Эти и многие другие пакеты можно найти на серверах CTAN, адреса которых приведены на стр. 8, или на компакт-диске, прилагаемом к части тиража нашей книги.

11.4.1. Пакет floatflt

Пакет floatflt Матса Далгрена (Dahlgren, Mats) вводит две процедуры

```
\begin{floatingfigure}[hpos]{width} ...\end{floatingfigure}
\begin{floatingtable}[hpos]{width}  ...\end{floatingtable}
```
(floatflt)

для размещения соответственно небольших рисунков и небольших таблиц заданной ширины *width*. Эти процедуры могут иметь необязательный аргумент *hpos*, управляющий размещением рисунка или таблицы. Рис. 11.5 во входном файле описан следующим образом:

```
\begin{floatingfigure}[l]{4.4cm}  % Вставляем рисунок
  \includegraphics*[width=4.0cm]{lion1}
  \caption{Мать Лая и Йона}\label{fig:5}
\end{floatingfigure}
```

В зависимости от чётности номера страницы, класса печатного документа и опций, с которыми загружен пакет floatflt, обтекаемые рисунки и таблицы могут располагаться с правой или левой стороны страницы. В нашем случае рисунок располагается по левому краю страницы вне зависимости от её номера, поскольку процедура `floatingfigure` вызвана с опцией `l`. Все возможные значения параметра *hpos* перечислены ниже:

`l` — разместить плавающий объект слева от абзаца;

`r` — разместить плавающий объект справа от абзаца;

`p` — разместить плавающий объект на внешнем крае страницы (справа для нечётной страницы, слева для чётной страницы).

Если необязательный аргумент *hpos* пропущен, способ позиционирования объекта определяется значением необязательного аргумента *defaultpos*, с котором загружен пакет floatflt:

`rflt` — разместить плавающие объекты справа;

`lflt` — разместить плавающие объекты слева.

По умолчанию, если и пакет, и процедура вызваны без необязательных аргументов, плавающий объект будет размещен по внешнему краю страницы.

Внутри процедур `floatingfigure` и `floatingtable` можно использовать команду `\caption`, которая создаёт подписи к рисункам. Только её присутствие делает различимыми эти процедуры. Процедура `floatingfigure` использует общую нумерацию плавающих объектов с процедурой `figure`, а `floatingtable` — с процедурой `table`.

Имея дело с нестандартными пакетами, нужно быть настороже. На собственном опыте мы убедились, что процедура `floatingfigure` иногда просто теряет рисунки, причём «эффект» может неожиданно появиться при удалении или добавлении одной строчки текста.

11.4.2. Пакет wrapfig

Пакет wrapfig Дональда Арсено (Arseneau, Donald) вводит процедуру

`\begin{wrapfigure}[`*nlines*`]{`*hpos*`}{`*width*`}` ... `\end{wrapfigure}` (wrapfig)

Два её аргумента *hpos* и *width* имеют тот же смысл, что и у процедур, которые вводит пакет floatflt, но в отличие от последних у процедуры `wrapfigure` оба эти аргумента обязательны. Необязательный аргумент *nline* также имеется и определяет число коротких строк, обтекающих рисунок. Этот аргумент следует задавать в том случае, если нужно подправить число коротких строк, подсчитанное автоматически. При подсчёте строк пакет `wrapfigure` отводит три строки математической формуле и добавляет длину промежутка

`\intextsep` (wrapfig)

Процедура `wrapfigure` оставляет зазор ширины

`\columnsep` (wrapfig)

между рисунком и соседними короткими строками.

Важное отличие от процедур `floatingfigure` и `floatingtable` состоит в том, что `wrapfigure` фактически создаёт объект, который «не плавает». Окончательный подбор размещения такого объекта приходится делать «вручную» перед окончанием работы над документом. Процедуру `wrapfigure` лучше всего размещать после завершения абзаца. Если же необходимо поместить её внутри абзаца, то это следует делать между словами в том месте, где естественным образом происходит переход на новую строку.

11.5. Заметки на полях

Заметку на полях страницы печатает команда

`\marginpar[l-text]{text}`

используя текст обязательного аргумента text, если заметка идёт на правое поле или если опущен необязательный аргумент l-text; содержимое последнего печатается в том случае, если заметка попадает на левое поле. Текст заметки форматируется в текстовой моде. На полях страницы заметка позиционируется так, чтобы её верхняя строка находилась на одном уровне со строкой текста, содержащей команду `\marginpar`. Если эта команда находится между абзацами, то заметка размещается на уровне последней строки предшествующего ей абзаца. Однако заметка смещается вниз (и соответствующее сообщение выводится на экран дисплея), если она перекрывает предыдущую заметку.

Это заметка на полях

Следующий пример показывает, как создана первая заметка на полях в данном разделе:

```
...первая заметка на полях\marginpar{\flushleft \small
    \emph{Это заметка\\ на полях}} в данном разделе...
```

Стандартные классы помещают текст из обязательного аргумента text на правое поле страницы при односторонней печати (когда чётные и нечётные страницы сформатированы одинаково). При двусторонней печати заметка помещается на внешнем (от переплёта) поле: на левом — для чётной, на правом — для нечётной страницы. Для класса `book` двусторонняя печать устанавливается по умолчанию, а для других классов — при наличии опции `twoside` в декларации `\documentclass`. При двухколоночной печати (опция `twocolumn`) заметка выносится на ближайшее поле. Эти правила, используемые по умолчанию, могут быть изменены следующими декларациями.

`\reversemarginpar` вводит обратный режим позиционирования: изменяет размещение заметок на полях на противоположное относительно используемого по умолчанию.

`\normalmarginpar` восстанавливает нормальный режим позиционирования, используемый по умолчанию.

Когда заметка на полях появляется внутри абзаца, её размещение регулируется декларацией, действующей непосредственно перед пустой строкой, завершающей абзац.

Вполне естественно, что текст заметки может зависеть от того, на правое или левое поле она попадёт. При наличии необязательного аргумента у команды `\marginpar` на левое поле пойдёт l-text, а не содержимое обязательного аргумента. В следующем примере

```
\marginpar[\hfill $\Rightarrow$]{$\Leftarrow$}
```

правая стрелка `$\Rightarrow$` ($\Rightarrow$) печатается, если заметка попадает на левое поле, а текст из обязательного аргумента `$\Leftarrow$` ($\Leftarrow$) — на правое поле. При любом исходе стрелка будет направлена на текст. Однако стрелки будут расположены несимметрично, поскольку будут смещены к левому краю пространства, зарезервированного для заметок на полях. Команда `\hfill` помогает отрегулировать их положение (раздел 4.3). $\Leftarrow$

Заметка на полях никогда не переносится по частям на следующую страницу. LaTeX смещает заметки на полях вниз, чтобы избежать их наложения друг на друга. Если заметка занимает более трёх строк, иногда приходится регулировать её расположение, перемещая команду `\marginpar` в тексте входного файла, пользуясь командами `\vspace` (раздел 4.6) либо регулируя место разбиения текста на страницы (раздел 4.7). В любом случае, это следует делать после завершения работы над всем документом в целом. LaTeX не очень-то хорошо умеет обращаться с заметками на полях. Поэтому, если их очень много, ему просто может не хватить памяти. В таком случае полезно вспомнить, что заметка — это то, что действительно стоит замечать!

11.5.1. Параметры настройки

`\marginparwidth` — ширина парбокса, содержащего заметку на полях. Нерастяжимая длина.

`\marginparsep` — горизонтальный пробел между внешней границей текста (границей страницы) и заметкой на полях. Нерастяжимая длина.

`\marginparpush` — минимальное вертикальное расстояние между двумя последовательными заметками на полях. Нерастяжимая длина.

См. также рис. 17.1 на стр. 402 в главе 17.

Табулятор — электромеханическая ЦВМ
для обработки числовых данных,
вводимых с перфокарты.
Энциклопедия Кирилла и Мефодия

Глава 12

Таблицы

Публикации в научно-технических и экономических изданиях часто сообщают массу деталей, оформленных в виде таблиц. LaTeX предлагает несколько процедур для представления табличного материала. Процедура tabbing аналогична печати таблиц на пишущей машинке. Она устанавливает положения табулятора, которые служат разделителями колонок. Более гибкая процедура tabular автоматически выбирает ширину колонок и позволяет проводить разделительные линии между строками и колонками. Аналогичная ей процедура array действует только в математической моде; её следует предпочесть, если бóльшую часть таблицы составляют математические формулы. Процедура array кратко описана в главе 6. Данная глава дополняет это описание и рассказывает ещё о нескольких процедурах, которые определены в пакетах array, dcolumn, delarray, hhline, longtable и tabularx.

Мы начнём изучение методов набора таблиц с основных процедур: tabbing и tabular. Процедура array мало чем отличается от tabular, а пакеты всего лишь расширяют возможности, скрытые в array и tabular, хотя и весьма значительно.

Главные различия между tabbing и tabular состоят в следующем.

- Процедуру tabbing можно использовать только в текстовом режиме: она создаёт абзац, состоящий из отдельных строк. Процедура tabular применима в любой моде: она создаёт таблицу в виде прямоугольного бокса, который можно поместить в середину формулы или строки текста. Поэтому с помощью процедуры tabular можно строить таблицы с очень сложной структурой, вкладывая одну процедуру в другую.

- LaTeX может построчно переносить на следующую страницу текст, обрабатываемый процедурой tabbing, но он не может начать новую страницу в середине текста, форматируемого процедурой tabular. Поэтому таблицу, созданную процедурой tabular, обычно размещают в виде плавающего объекта с помощью процедур table или figure (глава 11). Очень длинные таблицы печатают на нескольких страницах, используя пакет longtable.

- LaTeX автоматически устанавливает ширину колонок в tabular, но для процедуры tabbing Читатель должен сделать это сам, установив точки табуляции.

12.1. Процедура tabbing

```
\begin{tabbing}  ...  \end{tabbing}
```

Процедура `tabbing` разбивает текст на строки с выравниванием текста в колонках. Границами колонок служат точки табуляции. Табулятор установлен, если ему приписано расстояние от предыдущего табулятора. Самый левый (нулевой) табулятор всегда установлен там, где к началу процедуры `tabbing` находилась левая граница колонки текста. Точки табуляции устанавливаются командой `\=`, а команда `\>` передвигает текст к следующему (заранее установленному командой `\=`) положению табулятора. Строки разделяются командой `\\`.

Следующая таблица

Название	Жанр	*Автор*
Полтава	Стихи	Пушкин А. С.
Записки сумасшедшего	Проза	Гоголь Н. В.

во входном файле описана так:

```
\begin{tabbing}
AAAAAAAAAAAAAxxxxxxxx\=\hspace{9ex}\=              \kill
\itshape Название    \> Жанр       \> \itshape Автор\\
Полтава              \> Стихи      \> Пушкин А.\,С.   \\
Записки сумасшедшего \> Проза      \> Гоголь Н.\,В.
\end{tabbing}
```

Команда `\kill` в первой строке означает, что эту строку печатать не нужно, и она используется только для установки точек табуляции. Пробелы игнорируются после команд `\=` или `\>`, но не перед ними, причём десять пробелов по-прежнему равны одному. Длина строки перед `\=` определяет ширину устанавливаемой колонки. Команда `\hspace` во второй колонке приравняла её ширину девяти единицам длины `ex` (раздел 2.10). Если после `A` в первой колонке и в первой строке все буквы `x` заменить пробелами, то только один из них будет учтён при установке ширины колонки, и соседние колонки частично перекроют друг друга:

Название	Жанр	*Автор*
Полтава	Стихи	Пушкин А. С.
Записки сумасшедшего	Проза	Гоголь Н. В.

Интересно, что слово «Жанр» напечатано прямым шрифтом, хотя в предыдущей колонке имеется декларация `\itshape`. Дело в том, что команды табуляции одновременно ограничивают область действия любых деклараций точно так же, как фигурные скобки.

Позволим себе привести ещё один пример таблицы с колонками переменной ширины[1]:

[1] Мы намеренно сохраняем таблицу в неизменном виде, начиная с первого издания нашей книги, подготовленного к печати в середине 1993 года.

IBM PC	Дисплей	Память	Винчестер	
286/12MHz	VGA	0.64 MB	20MB	
		1 MB	40MB	
386/25MHz	VGA	1 MB	60MB	
386/25MHz	SVGA			комплектация по заказу
486/33MHz	SVGA	8 MB	660MB	

При её создании использована бóльшая часть существующих команд табуляции, а для расширения колонок использованы команды `\qquad` и `\␣`:

```
\begin{tabbing}
IBM PC\qquad \= Дисплей \ \= Память \ \  \= Винчестер \\
286/12MHz    \> VGA    \> \> 0.64 MB \' 20MB \+ \+    \\
                          \>    1 MB \' 40MB \-       \\
\< 386/25MHz \> VGA    \> \>    1 MB \' 60MB \-       \\
\pushtabs
386/25MHz \> SVGA \= \` комплектация по заказу        \\
\poptabs
486/33MHz \> SVGA     \> \>    8 MB \' 660MB
\end{tabbing}
```

Все перечисленные ниже команды, которые могут появляться внутри процедуры `tabbing`, являются хрупкими.

⚠ `\=` устанавливает положение табулятора в данном месте строки.

⚠ `\>` сдвигает текст к следующему табулятору.

⚠ `\\` начинает новую строку и сдвигает текст к табулятору левой границы, которым первоначально является нулевой табулятор.

⚠ `\kill` позволяет установить положение табуляторов без печати текста; действует аналогично команде `\\`, за тем исключением, что текущая строка не печатается; действие команд `\=`, `\+`, `\-` сохраняется.

⚠ `\+` сдвигает левую границу последующих строк на один табулятор вправо, как если бы в их начале была добавлена команда `\>`.

⚠ `\-` отменяет действие одной предшествующей команды `\+`, сдвигая левую границу последующих строк на один табулятор влево.

⚠ `\<` может использоваться только в начале строки: отменяет действие на эту строку одной предшествующей команды `\+`.

⚠ `\'` сдвигает весь предшествующий текст в текущей колонке, то есть всё, что напечатано после самой последней из команд `\>`, `\<`, `\'`, `\\`, `\kill`, к правой границе предыдущей колонки. Следующий за `\'` текст печатается, начиная с левого края текущей колонки (то есть от положения текущего табулятора).

⚠ `` \` `` сдвигает весь последующий текст в строке к правой границе страницы; после команды `` \` `` вплоть до конца строки не должно быть команд `\>`, `\=` или `\␣`.

⚠ `\pushtabs` сохраняет текущее положение табуляторов в процедуре `tabbing` для дальнейшего использования.

⚠ `\poptabs` восстанавливает табуляцию, предварительно сохранённую командой `\pushtabs`. Команды `\pushtabs` и `\poptabs` могут быть парами вложены друг в друга.

Процедуры `tabbing` не могут быть вложены друг в друга, даже если внутренняя процедура заключена в парбокс. Нужно также учитывать, что команды табуляции ограничивают область действия деклараций так же, как и фигурные скобки или `\end{tabbing}`. Любая другая процедура, содержащаяся внутри `tabbing`, не может содержать ни одну из этих команд.

Следует обратить внимание, что команды `\=`, `\'`, `` \` ``, `\-`, имеющие смысл вне процедуры `tabbing`, внутри неё переопределены. Первые три из них вне `tabbing` используются для печати символов с диакритическими знаками. Внутри `tabbing` соответствующие диакритические знаки печатают команды

⚠ `\a=`, `\a'`, `` \a` ``

Команда `\-` вне `tabbing` служит для указания точки переноса слова по слогам. Внутри `tabbing` перенос по слогам невозможен. Исходный смысл команд `\=`, `\'`, `` \` ``, `\-` восстанавливается внутри парбокса, если он содержится в теле процедуры `tabbing`.

12.1.1. Параметры настройки

`\tabbingsep` — расстояние между колонками.

12.2. Процедура tabular

Для форматирования сложных таблиц или схем чаще используют процедуру `tabular`. Она почти полностью аналогична процедуре `array`, описанной в главе 6, но в отличие от неё работает в любой моде, а не только в математической. Процедура `tabular` позволяет легко рисовать границы таблиц, проводить разделительные вертикальные и горизонтальные линии между их элементами. Строки в теле процедуры разделяются командой `\\` или `\tabularnewline`, а каждая строка обрабатывается в строковой моде, то есть не переносится автоматически на следующую строку, если не вмещается в текущую.

Следующая таблица

Романов	Пётр	император
Кутузов	Михаил	генерал
Курчатов	Игорь	физик

описана во входном файле так:

```
\begin{center}
    \begin{tabular}{|l|c||r|}           \hline
    Романов  & Пётр    & император   \\  \hline
    Кутузов  & Михаил  & генерал     \\  \hline
    Курчатов & Игорь   & физик       \\  \hline
    \end{tabular}
\end{center}
```

Здесь процедура `center` центрирует таблицу на странице по горизонтали. Сама таблица создана процедурой `tabular`. Её аргумент `{|l|c||r|}` называют *преамбулой таблицы*. Буквы `l`, `c`, `r` в преамбуле таблицы происходят от сокращения английских слов left, center, right. Они называются *указателями колонок* и означают, что в таблице должно быть три колонки, причём содержимое двух крайних должно быть выровнено соответственно по левому и правому краю колонок, а в средней колонке — расположено по центру. Символы `|` показывают, что между колонками нужно провести вертикальные линии на всю высоту таблицы. Горизонтальные линии проводит команда `\hline`.

Как процедура `tabular` может формировать колонки переменной ширины, показывает ещё один пример. Следующая пара таблиц

Imparfait			*Plus-que-parfait*		
j'	étais		j'	avais	été
tu	étais		j'	avais	été
il	était		j'	avait	été
nous	étions		nous	avions	été
vous	étiez		vous	aviez	été
ils	étaient		ils	avaient	été

создана одной процедурой `tabular`:

```
\begin{tabular}{|ll|l|lll|} \cline{1-2} \cline{4-6}
\multicolumn{2}{|c|}{\slshape Imparfait}   & \hspace{7mm} &
\multicolumn{3}{c|}{\slshape Plus-que-parfait} \\
\cline{1-2} \cline{4-6}
j'   & \'{e}tais   &      & j'   & avais   & \'{e}t\'{e}\\
tu   & \'{e}tais   &      & j'   & avais   & \'{e}t\'{e}\\
il   & \'{e}tait   &      & j'   & avait   & \'{e}t\'{e}\\
     &             &      &      &         &            \\
nous & \'{e}tions  &      & nous & avions  & \'{e}t\'{e}\\
vous & \'{e}tiez   &      & vous & aviez   & \'{e}t\'{e}\\
ils  & \'{e}taient &      & ils  & avaient & \'{e}t\'{e}\\
\cline{1-2} \cline{4-6}
\end{tabular}
```

Пробел длиной 7 мм между двумя частями таблицы в процедуре описан как пустая третья колонка единой таблицы. Ширина колонки определяется командой

`\hspace{7mm}` в первой строке исходного текста таблицы. Команды `\cline` проводят горизонтальные линии через колонки (в данном случае через колонки 1–2 и 4–6). Команда `\multicolumn` объединяет элементы нескольких колонок в одной строке в одну колонку, позволяя, как в данном примере, создать общий заголовок. Её первый аргумент показывает, сколько колонок нужно объединить. Второй аргумент замещает часть аргумента самой процедуры `tabular`, соответствующий объединённой колонке; он должен содержать одну и только одну букву `l`, `c` или `r` и, возможно, символы `|`. Третий аргумент собственно и есть текст в объединённой колонке.

Если аргумент процедуры содержит символы `|`, не совсем очевидно, какие из них будут замещены вторым аргументом команды `\multicolumn`. В этом случае действует правило, что часть аргумента процедуры, соответствующая отдельной колонке, начинается с буквы `l`, `c` или `r`. Поэтому в последнем примере второй аргумент у первой из двух команд `\multicolumn` начинается с символа `|`, а у второй — с буквы `c` (впрочем, в данном случае ничего бы не изменилось, если бы он также начинался с `|`).

Процедура `tabular` производит объект, называемый боксом (глава 9). Бокс, созданный процедурой `tabular`, можно вставить в середину абзаца или слова как обычную букву, но это будет выглядеть очень странно. Лучше его разместить в отдельной строке, например в её центре; именно это делает процедура `center`. LaTeX сам увеличит вертикальный размер строки до необходимой величины, но если высота строки превысит свободное расстояние до нижнего края страницы, LaTeX выдаст соответствующее предупреждение «`Overfull \vbox`»[2] и перенесёт бокс на новую страницу. Полностью проблему размещения больших боксов на странице решают процедуры `figure` и `table`, рассмотренные в главе 11.

Приведём теперь полное описание процедур типа `tabular`. Процедуры

`\begin{array}[`*vpos*`]{`*cols*`}`	*lines*	`\end{array}`
`\begin{tabular}[`*vpos*`]{`*cols*`}`	*lines*	`\end{tabular}`
`\begin{tabular*}{`*width*`}[`*vpos*`]{`*cols*`}`	*lines*	`\end{tabular*}`

производят бокс, состоящий из последовательности строк *lines*, разделённых на несколько ячеек, выровненных по вертикальным колонкам. Процедуры `tabular` и `tabular*` могут использоваться в любой моде, а процедура `array` только в математической. Назначение их аргументов описано ниже.

width указывает ширину бокса, создаваемого процедурой `tabular*`. Между колонками должен быть вставлен пробел растяжимой длины, чтобы растянуть таблицу до указанной ширины; см. описание команды `\extracolsep` ниже.

vpos указывает способ позиционирования таблицы в строке, где таблица будет напечатана; может принимать следующие значения:

- `t` — выравнивание по верхней строке (верхняя строка в таблице будет расположена на одном уровне со строкой, куда помещена таблица);

[2] Переполнение вертикального бокса.

`c` — выравнивание по центру таблицы (используется по умолчанию);

`b` — выравнивание по нижней строке.

cols указывает способ форматирования колонок и называется преамбулой таблицы. Состоит из последовательности перечисленных ниже указателей колонок и межколоночного материала:

`l` — колонка с выравниванием по левой границе;

`r` — колонка с выравниванием по правой границе;

`c` — колонка с выравниванием по центру;

`|` — вертикальная линия между колонками на всю высоту таблицы.

`@{`*text*`}` — вставка текста между колонками во все строки. Этот указатель называется @-выражением. Аргумент *text* @-выражения обрабатывается в математической моде в процедуре `array` и в строковой моде в процедурах `tabular` и `tabular*`; он рассматривается как подвижный аргумент, поэтому хрупкие команды внутри него должны быть защищены командой `\protect`.

@-выражение подавляет вертикальный пробел, который обычно вставляется между колонками; любой требуемый пробел между вставляемым текстом и соседними ячейками должен быть включён в *text*. Чтобы вместо стандартного расстояния между двумя колонками установить пробел длины *wd*, достаточно вставить команду `@{\hspace{`*wd*`}}` между соответствующими указателями колонок. Команда

`\extracolsep{`*wd*`}`

в @-выражении вставляет дополнительный пробел длины *wd* с левой стороны всех последующих колонок, пока его длина не будет изменена другой командой `\extracolsep` (однако она не вставляет пробел слева от первой колонки). В отличие от обычного межколоночного пробела, этот дополнительный пробел не подавляется @-выражением. Команда `\extracolsep` может быть использована только в @-выражении в аргументе *cols*. Обычно её используют, чтобы вставить пробел `\fill` между колонками в процедуре `tabular*`.

`p{`*wd*`}` — колонка, каждая *txt* ячейка которой форматируется в виде парбокса, как если бы он был аргументом команды `\parbox[t]{`*wd*`}{`*txt*`}` (раздел 9.2). Однако команда `\\` не может использоваться внутри любой такой ячейки, за исключением следующих ситуаций:

- внутри процедур типа `minipage`, `array` или `tabular`;
- внутри парбокса, явно указанного командой `\parbox`;
- в области действия деклараций `\centering`, `\raggedleft`, `\raggedright`, `\center`, `\flushright`, `\flushleft`, которые в этом случае должны находиться внутри фигурных или командных скобок.

`*{`*n*`}{`*cols*`}` — $n \geq 1$ копий колонок, где *cols* — любой список указателей, который может содержать другое `*`-выражение.

Дополнительный пробел, равный половине используемого по умолчанию межколоночного расстояния (раздел 12.2.1), вставляется перед первой колонкой (если *cols* не начинается с | или @-выражения) и за последней колонкой. Этот пробел обычно не создаёт проблем, но может быть легко устранён, если вставить `@{}` в начале и конце *cols*.

Тело процедуры состоит из последовательности строк, разделённых командами \\ или

`\tabularnewline[len]`

причём вторая из них введена специально для того, чтобы при желании во входном файле легче было отличить конец строки таблицы от конца строки в ячейке колонки `p` (см. выше). В остальном команда `\tabularnewline` эквивалентна `\\[len]`, но её можно использовать только для завершения строк в процедурах `tabular`, `tabular*`, `array`, а также в процедурах, определённых в пакетах tabularx и longtable из коллекции tools (разделы 12.4 и 12.5).

Каждая строка является последовательностью ячеек, разделённых символами `&`. Число ячеек должно быть не больше числа указателей колонок, специфицированных в преамбуле таблицы, т. е. в аргументе *cols*. Текст каждой ячейки обрабатывается так, как если бы он был заключён в фигурные скобки, поэтому область действия любой декларации, помещённой внутри ячейки, лежит внутри неё. Любая ячейка может включать следующие команды:

⚠ `\multicolumn{n}{col}{text}` создаёт ячейку, состоящую из текста *text*, занимающую *n* колонок и позиционированную в соответствии с *col*. При $n = 1$ эта команда используется для переопределения способа позиционирования, указанного в аргументе *cols* процедуры. Аргумент *col* команды должен содержать в точности один указатель `r`, `c`, `l` и одно или более @-выражение или символ(ы) `|`. Он замещает часть аргумента *cols* процедуры, соответствующую *n* замещаемым колонкам, причём часть, соответствующая отдельной колонке (за исключением первой), начинается с указателя `r`, `c`, `l` или `p`, так что `|c|l@{:}lr` имеет части `|c|`, `l@{:}`, `l` и `r`. Команда `\multicolumn` должна либо начинать строку, либо следовать непосредственно за `&`.

`\vline` проводит вертикальную линию на полную высоту и глубину строки. Команда `\hfill` (раздел 4.3) позволяет сдвинуть эту линию на границы колонки. Команда `\vline` может также использоваться в @-выражениях.

Следующие команды проводят горизонтальные линии. Они могут располагаться между строками, непосредственно вслед за \\, либо до первой строки, либо вслед за последней строкой, за которой в этом и только в этом случае нужно поставить команду \\:

⚠ `\hline` проводит горизонтальную линию на всю ширину бокса. Две последовательные команды `\hline` проводят две горизонтальные линии на некотором расстоянии друг от друга; вертикальные линии, созданные символом `|` в аргументе *cols*, не рисуются в этом промежутке.

⚠ `\cline{i–j}` проводит горизонтальную линию через колонки с порядковыми номерами от *i* до *j*. Две или более последовательные команды `\cline` проводят линии на одной и той же высоте. При определении материала, относящегося к конкретной колонке, следует помнить, что спецификатор колонки всегда начинается с буквы `c`, `r`, `l` или `p` (см. описание команды `\multicolumn`).

Следующий пример содержит таблицу с переменным числом колонок:

Животноводство			
	Цены		
Год	мин.	макс.	Примечания
1971	97–245		Неудачный год для фермеров на Западе
72	245–245		Уменьшение продаж из-за суровой зимы

Вторая и третья колонки в нижней части таблицы разделены символом –, поэтому числовой интервал 97–245 на самом деле размещён в двух колонках, а горизонтальное положение чисел в двух последних строках выровнено по черточке между ними. Для этого в аргументе процедуры `tabular` граница между второй и третьей колонками описана как `@{--}`:

```
\begin{tabular}{|r||r@{--}l|p{1.5in}|}                          \hline
  \multicolumn{4}{|c|}{Животноводство}                       \\ \hline\hline
      & \multicolumn{2}{c|}{Цены}
           &                                                 \\ \cline{2-3}
  \multicolumn{1}{|c||}{Год}
      & \multicolumn{1}{r@{\,\vline\,}}{мин.}
           & макс. & \multicolumn{1}{c|}{Примечания}       \\ \hline
  1971 & 97   & 245   & Неудачный год для фермеров на Западе\\ \hline
  72   & 245  & 245   & {\raggedright Уменьшение продаж\\
                           из-за суровой зимы}               \\ \hline
\end{tabular}
```

Четвёртая колонка описана как парбокс шириной 1,5 дюйма: `p{1.5in}`. Поэтому длинный текст в этой колонке автоматически разбивается на достаточное число строк. Так как в узких колонках трудно разбивать текст на строки без чрезмерного увеличения пробелов между словами, в последней ячейке последней колонки использована декларация `\raggedright`, разрешающая перенос слов без выравнивания правой границы колонки; область её действия явно выделена фигурными скобками.

Следующая таблица

	Таблица	
12345.6789	abcde	fghi:initial
1237.589	abch	efh:partial
6238.57941	abch	egida:full

← 65 мм →

имеет заданную ширину 65 мм. Она создана процедурой tabular*:

```
\begin{tabular*}{65mm}{|r@{.}l@{\extracolsep{\fill}}%
  cr@{\extracolsep{0pt}:}l|}
    \multicolumn{5}{c}{\underline{Таблица}}           \\
    12345&6789  & abcde    & fghi      & initial      \\
    1237 & 589  & abch     & efh       & partial      \\
    6238 &57941 & abch     & egida     & full
\end{tabular}
```

Здесь первая колонка отделена от второй десятичной точкой, а четвертая от пятой — двоеточием. Для достижения заданной ширины между второй и третьей колонками при помощи @{\extracolsep{\fill}} вставлена бесконечно растяжимая длина \fill. При этом должно бы также увеличиться расстояние между третьей и четвертой колонками, но ещё одна команда \extracolsep перед последней колонкой отменяет это увеличение.

Приведём ещё несколько простых примеров, показывающих, как можно управлять промежутками между текстом и границей колонки. Начнём с варианта, когда величина промежутков устанавливается по умолчанию (она определяется параметром \tabcolsep — см. ниже):

```
\begin{tabular}{|l|l|}              \hline
Текст1 & Текст2 \\ Текст3 & Текст4 \\ \hline
\end{tabular}
```

Текст1	Текст2
Текст3	Текст4

Здесь есть вертикальные линии между колонками, поэтому регулируемый зазор находится между текстом и вертикальной линией. Удалим сначала пробелы с внешних краев таблицы:

```
\begin{tabular}{|@{}l|l@{}|}   ...
```

Текст1	Текст2
Текст3	Текст4

А теперь с обеих сторон первой колонки:

```
\begin{tabular}{|@{}l@{}|l|}    ...
```

Текст1	Текст2
Текст3	Текст4

12.2.1. Параметры настройки

Следующие параметры могут быть изменены либо вне процедуры tabular, либо внутри отдельной ячейки. В первом случае изменения действуют на всю таблицу, во втором — область действия изменений ограничена имеющимися фигурными или командными скобками.

\arraycolsep — половина ширины горизонтального пробела между колонками в процедуре array; изменяется при помощи \setlength. Например, \setlength{\arraycolsep}{4pt} устанавливает расстояние между колонками равным 4 pt.

\tabcolsep — половина ширины горизонтального пробела между колонками в процедурах tabular и tabular*.

`\arrayrulewidth` — ширина линии, создаваемой `|` в аргументе *cols* процедуры, а также командами `\hline`, `\cline`, `\vline`.

`\doublerulesep` — ширина пробела между двойными линиями, создаваемыми двумя последовательными `|` или командами `\hline`.

`\arraystretch` — интервал между строками. Стандартное расстояние между строками, определяемое по высоте текста в строке, умножается на значение `\arraystretch`.

Для изменения значения `\arraystretch` (которое по умолчанию равно 1) следует использовать команду `\renewcommand`. Например, `\renewcommand{\arraystretch}{1.5}` увеличивает интервал в 1,5 раза. Другие перечисленные выше параметры имеют смысл длины и изменяются командами `\setlength` или `\addtolength`.

12.3. Пакет array

При подключении пакета array в процедурах `tabular`, `tabular*` и `array` появляются дополнительные инструменты настройки таблицы. Более того, становится реальностью создание новых указателей колонок, входящих в аргумент *cols* этих процедур.

Прежде всего перечислим новые и измененные указатели колонок.

`|` проводит вертикальную линию между колонками на всю высоту таблицы. В отличие от аналогичных процедур ядра LaTeX, расстояние между двумя колонками увеличено на толщину вертикальной линии. Это отличие особенно заметно для толстых линий.

`m{`*wd*`}` создаёт колонку, каждая ячейка которой *text* формируется в виде парбокса, как если бы он был аргументом команды `\parbox[c]{`*wd*`}{`*text*`}` (раздел 9.2), т. е. `m{`*wd*`}` действует аналогично `p{`*wd*`}`, но текст в элементах колонки будет центрироваться по высоте относительно текста в той же строке в соседних колонках.

`b{`*wd*`}` создаёт колонку, каждая ячейка которой *text* форматируется, как если бы он был аргументом команды `\parbox[b]{`*wd*`}{`*text*`}`. Последняя строка в элементах колонки будет расположена на одном уровне с соседними колонками.

`>{`*decl*`}` используется перед указателями `l`, `r`, `c`, `p`, `m` или `b`. Вставляет текст *decl* (чаще всего какую-нибудь декларацию) в начало каждой ячейки колонки.

`<{`*decl*`}` используется после указателей `l`, `r`, `c`, `p`, `m`, `b`. Вставляет текст *decl* точно в конец каждой ячейки колонки.

`!{`*text*`}` является аналогом указателя `|`, но вместо вертикальной линии вставляет между колонками *text*. В отличие от другого своего аналога `@{`*text*`}` не подавляет стандартный пробел между колонками.

Пакет array также вводит один новый параметр настройки. Это длина

`\extrarowheight` (array)

которая добавляется к нормальной высоте каждой строки таблицы (при неизменном вертикальном пробеле внизу строки).

Перейдём к примерам.

Чтобы текст в отдельной колонке был напечатан определённым шрифтом, достаточно поместить >{decl} с нужной декларацией decl перед указателем соответствующей колонки в аргументе cols процедуры `tabular`:

```
\begin{tabular}[t]{|c|>{\bfseries}l|>{\itshape}c|}
\hline A & B & C \\ \hline 100 & 10 & 1 \\ \hline
\end{tabular}
```

A	**B**	*C*
100	**10**	*1*

При этом отпадает необходимость вставлять одни и те же декларации во все элементы колонок. Увеличим теперь высоту всех строк на 3 pt, присвоив это значение параметру `\extrarowheight`:

```
\setlength{\extrarowheight}{3pt}
\begin{tabular}[t]{|c|>{\bfseries}l|>{\itshape}c|}
 ...
```

A	**B**	*C*
100	**10**	*1*

Если какая-нибудь колонка в процедуре `tabular` описана как `>{$}c<{$}`, то она автоматически будет форматироваться в математическом режиме. Напротив, эта же колонка в процедуре `array` будет обрабатываться в строковом (не математическом) режиме. И ещё немного экзотики. В следующем примере широкая вертикальная линия между колонками сконструирована с помощью указателя `\!`, команд `\vline` и `\setlength`:

```
\begin{tabular}
 {|c!{\setlength{\arrayrulewidth}{3pt}\vline}c|c|}
\hline A & B & C...
```

A	B	C
100	10	1

12.3.1. Создание новых указателей колонок

Вместо того чтобы постоянно набирать сложное описание колонки, можно ввести новый указатель колонки при помощи декларации

`\newcolumntype{`*x*`}[`*num*`]{`*def*`}` (array)

где *x* — имя нового указателя, *num* — число его аргументов, *def* — его определение (*num* и *def* имеют тот же смысл, что и в `\newcommand`). Например, для таблиц, где смешаны текст и математические формулы, будут полезны такие определения:

```
\newcolumntype{C}{>{$}c<{$}}
\newcolumntype{L}{>{$}l<{$}}
\newcolumntype{R}{>{$}r<{$}}
```

Тогда буква `C` в аргументе *cols* процедуры `tabular` будет предназначена для формул (которые уже не нужно будет окружать знаками `$`), размещаемых по центру колонки.

```
\begin{tabular}[t]{lLlL}
если & a>b, & то & |a-b|=a-b \\
если & a=b, & то & |a-b|=0   \\
если & a<b, & то & |a-b|=b-a
\end{tabular}
```

если	$a > b,$	то	$\lvert a-b\rvert = a-b$
если	$a = b,$	то	$\lvert a-b\rvert = 0$
если	$a < b,$	то	$\lvert a-b\rvert = b-a$

Определение *def* нового указателя *x* может содержать любые другие указатели, известные на момент использования указателя *x*. Выяснить, какие нестандартные указатели определены, поможет команда

`\showcols` (array)

Она выводит их определения на экран компьютера и в файл протокола компиляции *jobname*`.log`.

Один вновь введённый указатель может замещать несколько колонок одновременно. Например:

```
\newcolumntype{X}{lLlL}
\begin{tabular}{X}
если & a>b, & то...
```

если	$a > b,$	то	$\lvert a-b\rvert = a-b$
если	$a = b,$	то	$\lvert a-b\rvert = 0$
если	$a < b,$	то	$\lvert a-b\rvert = b-a$

Покажем ещё, как работают указатели с аргументами, повторив с очевидными изменениями последний пример из предыдущего раздела:

```
\newcolumntype{V}[1]
   {!{\setlength{\arrayrulewidth}{#1}\vline}}
\begin{tabular}{|cV{3pt}cV{0.4pt}c|}...
```

A	B	C
100	10	1

12.3.2. Пакет dcolumn

Пакет dcolumn вводит указатель колонки с тремя аргументами

`D{`*sep-tex*`}{`*sep-dvi*`}{`*int.dec*`}` (dcolumn)

который обеспечивает выравнивание чисел в колонке по десятичному знаку (запятой или точке). Его аргументы имеют следующий смысл:

sep-tex — символ, который будет использоваться в качестве десятичной точки во входном файле. Обычно этим символом является точка или запятая;

sep-dvi — символ, который будет использоваться в качестве десятичной точки в печатном документе. Обычно он совпадает с тем, что указан в первом аргументе *sep-tex*, но может быть также любым математическим выражением, как, например, `\cdot`;

int.dec — десятичное число, определяющее формат колонки, причём *int* задаёт количество цифр слева, а *dec* — справа от десятичной точки. Число с указанным количеством цифр будет размещено по центру колонки, а все числа иного сорта будут выровнены относительного него так, чтобы десятичные знаки располагались на одной вертикали. Целая часть *int.* (вместе с точкой) может быть опущена. В этом случае *dec* указывает максимальное количество цифр после десятичного знака. Отрицательное значение *dec* разрешает любое количество символов в колонке, а все числа в колонке центрируются по десятичному знаку.

Разумеется, можно ввести более простой указатель колонки, использующий свойства указателя `D`. Например, колонка типа `d`, определённая как

```
\newcolumntype{d}[1]{D{.}{\cdot}{#1}}
```

в качестве единственного аргумента принимает максимальное число цифр после десятичного знака, а колонки типа `.` и `,` вообще не имеют аргументов:

```
\newcolumntype{.}{D{.}{.}{-1}}
\newcolumntype{,}{D{,}{,}{2}}
```

В колонках `d` десятичная точка будет расположена выше базисной линии, поскольку её печатает команда `\cdot`; в двух других колонках десятичная точка будет иметь соответственно вид точки и запятой, как показывает следующий пример:

```
\begin{tabular}{|d{-1}|d{2}|.|,|}
1.2    & 1.2    &1.2     &1,2     \\
1121.2& 1121.2&861.20 &674,29 \\
184    & 184    &10      &69      \\
.4     & .4     &.4      &,4
\end{tabular}
```

1·2	1·2	1.2	1,2
1121·2	1121·2	861.20	674,29
184	184	10	69
·4	·4	.4	,4

Здесь первая колонка, имеющая отрицательное значение *dec*, шире второй колонки, так как десятичная точка расположена в её центре, а во второй прижата к левому краю. Возможно, однако, что указание только *dec* вместо полной спецификации *int.dec* не приводит к желаемому результату. Такое случается, если колонки таблицы имеют заголовки, созданные при помощи `\multicolumn{1}{c}{...}`, чтобы переопределить тип колонки. Следующий пример показывает действие полной спецификации *int.dec* в том случае, когда заголовки колонок у́же числовых данных:

```
\begin{tabular}[t]{|d{-1}|d{1}|d{5.1}|}
\multicolumn{1}{|c|}{head}&
\multicolumn{1}{c|}{head}&
\multicolumn{1}{c|}{head}     \\[3pt]
 1.2      & 1.2      & 1.2      \\
 11212.2 & 11212.2 & 11212.2  \\
 .4       & .4       &.4
\end{tabular}
```

head	head	head
1·2	1·2	1·2
11212·2	11212·2	11212·2
·4	·4	·4

Здесь предполагается, что для указателя `d` действует определение, данное в начале этого раздела, так что запись `d{-1}` эквивалентна `D{.}{\cdot}{-1}`. Колонки, описанные как `d{1}` и `d{5.1}` имеют одинаковую ширину, потому что самая длинная целая часть чисел в колонках как раз состоит из пяти цифр, а десятичная часть — из одной. В другом примере заголовки колонок шире чисел:

```
\begin{tabular}[t]{|d{-1}|d{1}|d{1.1}|}
 \multicolumn{1}{|c|}{heading}&
 \multicolumn{1}{c|}{heading}&
 . . .
```

heading	heading	heading
1·2	1·2	1·2
·4	·4	·4

Числа в третьей колонке `d{1.1}` центрируются по десятичной точке (как и в первой колонке `d{-1}`), так как значение `1.1` аргумента *int.dec* означает, что целая и десятичная части чисел имеют равную длину.

12.3.3. Таблицы в таблицах

Если таблица, созданная процедурой типа `tabular`, размещена в строке текста, то по умолчанию она центрируется по высоте, но может быть позиционирована вровень с верхней строкой, если необязательный аргумент *vpos* имеет значение `t`, или нижней, если *vpos* имеет значение `b` (см. стр. 285). Однако если таблица начинается с горизонтальной черты или заканчивается ею, то выравнивание осуществляется именно по этой линии.

```
Сравните таблицу
    \begin{tabular}[t]{|l|}
        без команды \\ \verb|\hline|
    \end{tabular}
и таблицу
    \begin{tabular}[t]{|l|}           \hline
       с командой \\ \verb|\hline| \\ \hline
    \end{tabular} на одной строке.
```

Сравните таблицу | без команды `\hline` | и таблицу | с командой `\hline` | на одной строке.

Чтобы даже в этом случае выравнивались именно строки, пакет array вводит две специальные команды

`\firsthline    \lasthline` (array)

предназначенные для рисования горизонтальной черты перед первой строкой и после последней строки таблицы соответственно.

```
Сравните таблицу
    \begin{tabular}[t]{|l|}
        без команды\\ \verb|\firsthline|
```

```
    \end{tabular}
и таблицу
    \begin{tabular}[t]{|l|}                 \firsthline
        с командой\\ \verb|\firsthline| \\  \lasthline
    \end{tabular} на одной строке.
```

Сравните таблицу | без команды \firsthline | и таблицу | с командой \firsthline | на одной строке.

Эти две команды вставляют дополнительный вертикальный пробел между горизонтальной чертой и, соответственно, первой и последней строками таблицы. Величина пробела задаётся командной длиной

`\extratabsurround` (array)

Регулировка этих пробелов таблицы особенно полезна при создании сложных таблиц при помощи нескольких процедур типа `tabular`, вложенных одна в другую. Пример такой таблицы приведён на рис. 12.1. Во входном файле она записана следующим образом:

```
\setlength{\extratabsurround}{2pt}
\begin{tabular}{|cc|}    \hline
\textit{Имя} &  \textit{телефон}                      \\ \hline\hline
 Джон &
  \begin{tabular}[t]{|cc|}                              \firsthline
  \textit{день} & \multicolumn{1}{c|}{\textit{телефон}}
                                                       \\ \hline\hline
  среда      & 5554434                                 \\ \hline
  понедельник &
     \begin{tabular}[t]{|cc|}                           \firsthline
      \textit{время}   & \textit{телефон}              \\ \hline\hline
       8--10 & 5520104 \\ 1--5 & 2425588               \\ \lasthline
     \end{tabular}                                     \\ \lasthline
  \end{tabular}                                        \\ \hline
 Мартин &
     \begin{tabular}[t]{|cp{4.5cm}|}                    \firsthline
      \textit{телефон} & \multicolumn{1}{c|}{\itshape инструкции}
                                                       \\ \hline\hline
      3356677 & Маша должна передать сообщение         \\ \lasthline
     \end{tabular}                                     \\ \hline
 Петя   &
     \begin{tabular}[t]{|cl|}                           \firsthline
      \textit{месяц} & \multicolumn{1}{c|}{\itshape телефон}
                                                       \\ \hline\hline
       сентябрь--май & 5554434 \\ июнь & нет телефона \\
       июль--август  & 2211456                         \\ \lasthline
     \end{tabular}                                     \\ \hline
 \end{tabular}
```

<table>
<tr><td>Имя</td><td>телефон</td></tr>
<tr><td>Джон</td><td>
<table>
<tr><td>день</td><td>телефон</td></tr>
<tr><td>среда</td><td>5554434</td></tr>
<tr><td>понедельник</td><td>
<table>
<tr><td>время</td><td>телефон</td></tr>
<tr><td>8–10</td><td>5520104</td></tr>
<tr><td>1–5</td><td>2425588</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Мартин</td><td>
<table>
<tr><td>телефон</td><td>инструкции</td></tr>
<tr><td>3356677</td><td>Маша должна передать сообщение</td></tr>
</table>
</td></tr>
<tr><td>Петя</td><td>
<table>
<tr><td>месяц</td><td>телефон</td></tr>
<tr><td>сентябрь–май</td><td>5554434</td></tr>
<tr><td>июнь</td><td>нет телефона</td></tr>
<tr><td>июль–август</td><td>2211456</td></tr>
</table>
</td></tr>
</table>

Рис. 12.1. Пример сложной таблицы

12.3.4. Пакет hhline

Пакет hhline вводит команду

`\hhline{cross}` (hhline)

которая рисует горизонтальные линии в таблицах. Её аргумент cross должен содержать последовательность особых символов, определяющих вид горизонтальной линии в каждой колонке и способ пересечения горизонтальных линий с вертикальными.

```
\begin{tabular}{||cc||c|c||}
 \hhline{|t:==:==:t|}
 a & b & c & d \\ \hhline{|:==:|~|~||}
 1 & 2 & 3 & 4 \\ \hhline{#==#~|=#}
 i & j & k & l \\ \hhline{||--||--||}
 w & x & y & z \\ \hhline{|b:==:b:==:b|}
\end{tabular}
```

a	b	c	d
1	2	3	4
i	j	k	l
w	x	y	z

Этот пример демонстрирует все возможности команды `\hhline`, и нам остаётся только пояснить значение символов, которые могут присутствовать в её аргументе.

- `-` Горизонтальная линия на всю ширину колонки.
- `=` Двойная горизонтальная линия на всю ширину колонки.
- `~` Колонка без горизонтальной линии.

`|` Пересечение вертикальной линии с горизонтальной (одинарной или двойной).

`#` Пересечение двойной вертикальной линии с двойной горизонтальной линией.

`:` Вертикальная линия, которая прерывается двойной горизонтальной линией.

`t` Верхняя половина двойной горизонтальной линии.

`b` Нижняя половина двойной горизонтальной линии.

`*{`n`}{`$cross$`}` n-кратное дублирование $cross$; например, `*{3}{==#}` эквивалентно `==#==#==#`.

12.3.5. Пакет delarray

Пакет delarray расширяет синтаксис процедуры `array`, позволяя удобным способом указывать, какими скобками нужно окружить матрицу. Например, если нужно заключить матрицу в круглые скобки, достаточно это проделать с аргументом процедуры (вместе с фигурными скобками!):

```
\[ \begin{array}({cc}) a & b \\ c & d \end{array} \]
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Скобки (точнее — разделители) всегда должны использоваться парами, как если бы команды `\left` и `\right`, при помощи которых формируются большие разделители (раздел 6.3.6), были указаны явно. Если один разделитель не нужен, его заменяют точкой.

```
\newcolumntype{L}{>{$}l<{$}}
\[ f(x)=\begin{array}\{{lL}.
        1,         & если  $x=0$\\
        \sin(x)/x & иначе
        \end{array}              \]
```

$$f(x) = \left\{ \begin{array}{ll} 1, & \text{если } x = 0 \\ \sin(x)/x & \text{иначе} \end{array} \right.$$

Пакет delarray автоматически загружает пакет array и использует заложенные в нём возможности (в частности, `\newcolumntype`). Однако не нужно бояться заказать один и тот же пакет дважды, так как компилятор загружает любой пакет только однажды.

Следующий пример показывает, что между традиционным способом генерации больших скобок при помощи `\left(`...`\right)` и тем, который вводит пакет delarray, есть некоторые различия. Они проявляются, если процедура `array` вызвана с опцией `t` или `b`:

$$\begin{pmatrix}1\\2\\3\end{pmatrix} \begin{pmatrix}4\\5\\6\end{pmatrix} \begin{pmatrix}7\\8\\9\end{pmatrix} \neq \left(\begin{array}[b]{c}1\\2\\3\end{array}\right) \begin{pmatrix}4\\5\\6\end{pmatrix} \left(\begin{array}[t]{c}7\\8\\9\end{array}\right)$$

Левая часть этого выражения набрана при помощи круглых скобок вокруг аргумента процедуры `array`:

```
\begin{array}[t]({c}) 1\\2\\3 \end{array}
\begin{array}[c]({c}) 4\\5\\6 \end{array}
\begin{array}[b]({c}) 7\\8\\9 \end{array}
```

В правой части использованы большие разделители `\left(` и `\right)`:

```
\left(\begin{array}[t]{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}[c]{c} 4\\5\\6 \end{array}\right)
\left(\begin{array}[b]{c} 7\\8\\9 \end{array}\right)
```

Выбор решения зависит от желаемого результата.

12.4. Пакет tabularx

Пакет tabularx вводит процедуру `tabularx`:

```
\begin{tabularx}{width}[vpos]{cols}
lines
\end{tabularx}
```
(tabularx)

Она аналогична процедуре `tabular*` и имеет те же аргументы. Напомним, что `tabular*` создаёт таблицу заданной ширины *width*, раздвигая при необходимости колонки. Процедура `tabularx` достигает той же цели, увеличивая ширину колонок. Колонки, подлежащие расширению, маркируются в аргументе *cols* специальным указателем `X`. Процедура конвертирует этот указатель в `p{`*wd*`}`, вычисляя при этом ширину колонки *wd*. Например, `\begin{tabularx}{100mm}{|X|X|X|}` задаёт таблицу шириной 100 миллиметров с тремя колонками одинаковой ширины.

Как и в пакете array, способ форматирования колонки `X` можно модифицировать при помощи указателя `>{`*decl*`}`. Например, в колонке `>{\small}X` будет использован шрифт уменьшенного размера. Другой формат, часто используемый в узких колонках, есть выравнивание текста по левому краю. Однако очевидное, казалось бы, решение `>{\raggedright}X` не достигает цели, так как команда `\\` не может одновременно разделять строки в колонке и строки в таблице. Поэтому пакет tabularx вводит команду

```
\arraybackslash
```
(tabularx)

для разделения строк в узких колонках. Именно её следует использовать после деклараций `\raggedright`, `\raggedleft` и `\centering` в колонке `X`. Например, автоматическое выравнивание влево достигается, если растяжимую колонку описать как `>{\raggedright\arraybackslash}X`. Для таких колонок можно ввести новый указатель, например

```
\newcolumntype{Y}{>{\small\raggedright\arraybackslash}X}
```

и затем использовать наравне с `X`.

```
\setlength{\extratabsurround}{1pt}
\begin{tabularx}{160pt}{|c|Y|X|}   \hline
\multicolumn{2}{|c|}{1 и 2} & 3 \\ \hline
1 & Ширина колонки зависит от
    ширины таблицы
    & Колонки 2 и 3 имеют
      одинаковую ширину        \\ \hline
\end{tabularx}
```

1 и 2		3
1	Ширина колонки зависит от ширины таблицы	Колонки 2 и 3 имеют одинаковую ширину

Элементы таблицы в колонке X позиционируются по верхней строке (поскольку это колонка типа p). Чтобы изменить способ позиционирования (сделать X колонкой типа m или b), необходимо переопределить команду

`\tabularxcolumn{wd}` (tabularx)

Значение аргумента wd будет вычислено процедурой tabularx. Указанная команда используется для построения колонок X. Её начальное определение эквивалентно

```
\newcommand{\tabularxcolumn}[1]{p{#1}}
```

где p — упомянутый выше указатель колонки. Возможно альтернативное переопределение

```
\renewcommand{\tabularxcolumn}[1]{>{\small}m{#1}}
```

Тогда элементы колонки X будут центрироваться по высоте, а размер шрифта будет уменьшен.

Все колонки X в таблице имеют одинаковую ширину. Чтобы сформировать колонки разной ширины, следует переопределить командную длину

`\hsize` (tabularx)

Например, описание

```
>{\setlength{\hsize}{.8\hsize}}X>{\setlength{\hsize}{1.2\hsize}}X
```

в аргументе cols процедуры tabularx задаёт две колонки, причём вторая в полтора раза шире первой. Формируя колонки разной ширины, следует сохранять неизменной суммарную ширину колонок (в приведённом выше примере она равна 2\hsize); кроме того, нельзя использовать команду \multicolumn для объединения колонок X неравной ширины. Если процедура tabularx используется внутри другой процедуры из семейства tabular, она должна быть окружена фигурными скобками (для других процедур построения таблиц такие предосторожности излишни).

При вычислении ширины колонок процедура tabularx делает несколько итераций, пытаясь выбрать наиболее оптимальные размеры. Понаблюдать за результатами её промежуточных вычислений можно, вставив во входной файл (например, в его преамбулу) декларацию

`\tracingtabularx` (tabularx)

Результат вычислений выводится на экран и в файл протокола. Тот же эффект достигается, если пакет tabularx загрузить с опцией `debugshow` или `infoshow` (в `\usepackage` или `\documentclass`).

12.5. Пакет longtable

Пакет longtable вводит одноимённую процедуру

```
\begin{longtable}[hpos]{cols}
lines
\end{longtable}
```
(longtable)

Сохраняя все свойства процедуры `tabular` (в её расширенном варианте, если загружен ещё пакет array), она предназначена, главным образом, для печати очень больших таблиц, не умещающихся на одной странице. Обязательный аргумент *cols* имеет тот же смысл, что и в процедуре `tabular`. Напротив, опция *hpos* определяет позиционирование длинной таблицы по горизонтали, тогда как необязательный аргумент *vpos* в процедуре `tabular` определяет позиционирование таблицы по вертикали в текущей строке. Соответственно этому *hpos* может принимать значения `l`, `c`, `r`, но не `t` или `b`. Опция `l` означает, что таблица будет прижата к левому краю страницы, `r` — к правому, а `c` означает, что таблица будет расположена по центру страницы. Правило позиционирования длинной таблицы по умолчанию (когда `[`*hpos*`]` отсутствует) соответствует опции `c`, но его можно существенно модифицировать. Например, легко сделать так, чтобы таблица отстояла от левого или правого края страницы на заданное расстояние (см. ниже).

Поскольку процедура `longtable` сама заботится о том, куда лучше всего поместить таблицу, она обладает также некоторыми свойствами процедуры `table`, описанной в главе 11 и используемой для размещения таблиц, умещающихся на одной странице. В частности, в процедуре `longtable`, как и в `table`, можно формировать подписи к таблице при помощи команды `\caption`, которая одновременно используется для нумерации таблиц. При этом и длинные (`longtable`), и обычные (`table`) таблицы будут пронумерованы при помощи единого счётчика `table`. Список всех таблиц печатает команда `\listoftables` (раздел 3.10).

Алгоритм работы процедуры `longtable` основывается на двух принципах.

Первый: чтобы не перегружать память компьютера, процедура строит большую таблицу, разбивая её на небольшие части. Их размер определяется значением счётчика

`LTchunksize` (longtable)

и по умолчанию составляет двадцать строк.

Второй принцип является следствием первого. При первом проходе компилятор выбирает ширину колонок в пределах каждой части независимо от ширины колонок в пределах последующих частей. Поэтому входной файл, вообще

говоря, необходимо обработать несколько раз, чтобы выровнять ширины колонок во всей таблице. При каждом проходе процедура вычисляет ширину колонок во всех частях и записывает необходимую информацию во вспомогательный файл (с расширением `aux`). Алгоритм сходится тем быстрее, чем больше величина `LTchunksize`, поэтому при наличии резерва памяти у компьютера значение счётчика полезно увеличить. В сложных случаях для получения окончательного результата необходимо обработать входной файл несколько раз. Компилятор выдаёт предупреждение о необходимости повторить обработку, выводя на экран (и в файл протокола) предупреждение

`Package longtable Warning: Table widths have changed. Rerun LaTeX`[3].

Таблица, изображённая на рис. 12.2, обработана со значением `LTchunksize` в два раза меньше принятого по умолчанию, но алгоритм построения таблицы сошелся за два прохода. При наличии нескольких команд `\multicolumn` меньше двух проходов и не бывает.

Текст соответствующего входного файла приведён ниже:

```
\documentclass{article}
\usepackage[cp1251]{inputenc}
\usepackage[english,russian]{babel}
\usepackage{array,longtable}
\setcounter{LTchunksize}{10}
 ...
\begin{document}
\begin{longtable}[c]{|*{3}{>{\ttfamily}cl}|}
\caption{Коды языков народов мира}\label{t:LongTable}          \\ \hline
\multicolumn{6}{|c|}{Коды языков (ISO 639:1988)}               \\
\textrm{код}&язык&\textrm{код}&язык&\textrm{код}&язык          \\ \hline
                                                  \endfirsthead   \hline
\multicolumn{6}{|c|}{\small\slshape (продолжение)}             \\ \hline
\textrm{код}&язык&\textrm{код}&язык&\textrm{код}&язык          \\ \hline
                                                  \endhead        \hline
\multicolumn{6}{|r|}{\small\slshape продолжение следует}       \\ \hline
                                                  \endfoot        \hline
                                                  \endlastfoot
aa & Afar          & ab & Abhaszian       & af & Afrikaans   \\
am & Amharic       & ar & Arabic          & as & Assamese    \\
  . . . . . . . . . . . . . . . . . . . . . . . . .
  . . . . . . . . . . . . . . . . . . . . . . . . .
lt & Lithuanian    & lv & Latvian, Lettish &  &             \\
\newpage
mg & Malagasy      & mi & Maori           & mk & Macedonian  \\
  . . . . . . . . . . . . . . . . . . . . . . . . .
  . . . . . . . . . . . . . . . . . . . . . . . . .
zh & Chinese       & zu & Zulu            &    &             \\
```

[3] Предупреждение пакета longtable: Ширина таблицы изменилась. Запустите LaTeX ещё раз.

Таблица 1: Коды языков народов мира

Коды языков (ISO 639:1988)					
код	язык	код	язык	код	язык
aa	Afar	ab	Abkhaszian	af	Afrikaans
am	Amharic	ar	Arabic	as	Assamese
ay	Aymara	az	Azerbaijan		
ba	Bashkir	be	Byelorussian	bg	Bulgarian
bh	Bihari	bi	Bislama	bn	Bengsli, Bangla
bo	Tibetan	br	Breton		
ca	Catalan	co	Corsican	cs	Czech
cy	Welch				
da	Danish	de	German	dz	Bhutani
el	Greek	en	English	so	Esperanto
es	Spanish	et	Estonian	eu	Basque
fa	Persian	fi	Finnish	fj	Fiji
fo	Faeroese	fr	French	fy	Frisian
ga	Irish	gd	Scots Gaelic	gl	Galician
gn	Guarani	gu	Gujarati		
ha	Hausa	hi	Hindi	hr	Croatian
hu	Hungarian	hy	Armenian		
ia	Interlingua	ie	Interlingue	ik	Inupiak
in	Indonesian	is	Icelandic	it	Italian
iw	Hebrew				
ja	Japanese	ji	Yiddish	jw	Javanese
ka	Georgian	kk	Kazakh	kl	Grenlandic
km	Cambodian	kn	Kannada	ko	Korean
ks	Kashmiri	ku	Kurdish	ky	Kirghiz
la	Latin	ln	Lindala	lo	Laothian
lt	Lithuanian	lv	Latvian, Lettish		
					продолжение следует

(продолжение)					
код	язык	код	язык	код	язык
mg	Malagasy	mi	Maori	mk	Macedonian
ml	Malayalam	mn	Mongolian	mo	Moldavian
mr	Marathi	ms	Malay	mt	Maltese
my	Burmese				
na	Nauru	ne	Nepali	nl	Dutch
no	Norwegian				
oc	Occitan	om	(Afan) Oromo	or	Oriya
pa	Punjabi	pl	Polish	ps	Pashto, Pushto
pt	Portuguese				
qu	Quechua				
rm	Rhaeto-Romanse	rn	Kirundi	ro	Romanian
ru	Russian	rw	Kinyarwanda		
sa	Sanskrit	sd	Sindhi	sq	Sangro
sh	Serbo-Croatian	si	Singhalese	sk	Slovak
sl	Slovenian	sm	Samoan	sn	Shona
so	Somali	sq	Albanian	sr	Serbian
ss	Siswati	st	Sesotho	su	Sudanese
sv	Swedish	sw	Swahili		
ta	Tamil	te	Tegulu	tg	Tajik
th	Thai	ti	Tigrinya	tk	Turkmen
tl	Tagalog	tn	Setswana	to	Tonga
tr	Turkish	ts	Tsonga	tt	Tatar
tw	Twi				
uk	Ukrainian	ur	Urdu	uz	Uzbek
vi	Vietnamese	vo	Volapuk		
wo	Wolof				
xh	Xhosa				
yo	Yoruba				
zh	Chinese	zu	Zulu		

Рис. 12.2. Пример длинной таблицы, размещённой на двух страницах. Изображение уменьшено и повёрнуто на 90°. Формат заголовка таблицы установлен классом `article` и поэтому отличается от рекомендованного отечественными типографскими правилами

```
\end{longtable}
\end{document}
```

Помимо longtable здесь загружен также пакет array, чтобы можно было использовать расширенный синтаксис указателей колонок, и в частности `>{\ttfamily}c`.

Таблица начинается с главного заголовка. В нашем примере он состоит из трёх строк и заканчивается командой

`\endfirsthead` (longtable)

Она завершает текущую строку так же, как и команда `\\`, но ещё обозначает конец заголовка. Заголовок таблицы на первой странице обычно отличается от заголовка на последующих страницах. Описание последнего отделяется от последующего текста командой

`\endhead` (longtable)

В нашем примере он состоит из двух строк. Помимо заголовков, таблица может иметь нижние колонтитулы. Их описание в нашем примере следует за описанием заголовков. Стандартный колонтитул печатается на всех страницах, кроме последней. Он заканчивается командой

`\endfoot` (longtable)

Описание нижнего колонтитула на последней странице завершает команда

`\endlastfoot` (longtable)

В нашем примере этот колонтитул пуст, так что команду `\endlastfoot` можно было опустить. Вслед за описанием заголовка и нижнего колонтитула идёт основное содержание таблицы, в котором нет ничего необычного. Заметим, что величина счётчика `LTchunksize` не может быть меньше числа строк в заголовках или колонтитулах, то есть для рассматриваемого примера она не должна быть меньше трёх.

Возвращаемся к главному заголовку. Его первая строка содержит команду `\caption`. Она печатает порядковый номер таблицы и первую строку её заголовка. Вообще говоря, заголовок оформляется так же, как любая другая часть таблицы, формируемой процедурой `longtable`, а команда `\caption` вовсе не является исключительной собственностью заголовка и может появляться в любой другой части таблицы. Полный синтаксис команды включает два аргумента и `*`-форму (отсутствующую в формате LaTeX):

`\caption[`*brief-caption*`]{`*full-caption*`}`
`\caption*{`*full-caption*`}` (longtable)

Здесь *full-caption* — текст заголовка. Если необязательный аргумент отсутствует, то *full-caption* попадёт также в список таблиц, который печатает команда `\listoftables`, иначе в список направляется *brief-caption*. Если `\caption{...}`

находится в повторяющемся заголовке или нижнем колонтитуле, то все эти заголовки и колонтитулы будут пронумерованы (с нарастающим значением порядкового номера) и внесены в список таблиц. Если такой исход нежелателен, следует использовать команду `\caption*{...}`, которая не печатает номер и ничего не вносит в список таблиц, или команду `\caption[]{...}` (с пустым необязательным аргументом), которая хотя и нумерует таблицу, но не вносит её в список.

После команды `\caption` (или в её аргумент) следует вставить команду `\label{`*key*`}`, чтобы затем иметь возможность сделать ссылку на таблицу при помощи команд `\ref{`*key*`}` и `\pageref{`*key*`}`. По понятным причинам нельзя вставлять `\label` в повторяющийся заголовок или колонтитул — компилятор запутается, обнаружив несколько пронумерованных объектов с одной и той же меткой.

В длинной таблице можно использовать команды `\footnote` для подстрочного примечания и `\newpage` для принудительного перехода на новую страницу. Последняя может пригодиться в колонках, соответствующих указателям `p`, `m` и `b`, так как процедура `longtable` ни за что не разорвёт таблицу внутри элемента такой колонки, даже если тот состоит из нескольких строк. Новая страница начинается между строками таблицы. Если строки разделяются горизонтальными линиями `\hline`, то линии автоматически дублируются при переносе на другую страницу. Команду `\footnote` нельзя использовать в повторяющихся частях таблицы по той же причине, что и команду `\label`. Чтобы сделать подстрочное примечание, например, к повторяющемуся заголовку, нужно в него вставить `\footnotemark` вместо `\footnote`. Напомним, что `\footnotemark` печатает метку подстрочного примечания, а сам текст примечания печатает команда `\footnotetext`; её нужно поместить куда-нибудь в ту часть таблицы, которая попадает на нужную страницу. Впрочем, эти сложности лишь подтверждают эмпирическое правило: то, что LaTeX трудно заставить сделать, обычно есть признак плохого стиля.

Теперь вернёмся в начало примера и обратим внимание Читателя на то, что процедура `longtable` вызвана без опции `[`*hpos*`]`. Это означает, что таблица должна быть расположена в центре страницы (по горизонтали). Однако это правило не абсолютно и его легко изменить. В том случае, когда опция `[`*hpos*`]` в `\begin{longtable}` отсутствует, позиционирование таблицы по горизонтали зависит от того, как определены командные длины

`\LTleft   \LTright` (longtable)

Слева от таблицы вставляется пробел ширины `\LTleft`, а справа — пробел ширины `\LTright`. Первоначально обе команды `\LTleft` и `\LTright` определены как `\fill`, поэтому таблица центрируется. Они могут быть переопределены. Например, определение

```
\setlength{\LTleft}{\parindent}
```

(при исходном значении `\LTright`) означает, что таблица будет размещена на расстоянии стандартного абзацного отступа от левого края предшествующего таблице текста. Экспериментируя с длинами `\LTleft` и `\LTright`, нужно помнить,

что по крайней мере одна из них должна быть растяжимой длиной, чтобы можно было заполнить всю ширину страницы, если сама таблица не может быть растянута за счёт увеличения межколоночного расстояния при помощи `\extracolsep`. Чтобы напечатать таблицу из двух колонок в полную ширину страницы, нужно делать примерно так:

```
\setlength{\LTleft}{0pt}
\setlength{\LTright}{0pt}
\begin{longtable}{l@{\extracolsep{\fill}}l}
```

Расстояние перед таблицей и после неё определяется командами

`\LTpre \LTpost` (longtable)

Это растяжимые длины; по умолчанию они равны `\bigskipamount`. Ширина подписей к таблице, печатаемых при помощи `\caption`, определяется командной длиной

`\LTcapwidth` (longtable)

Первоначально она равна 4 дюймам.

Процедура `longtable` начинает печатать длинную таблицу с новой строки на текущей странице. Чтобы начать таблицу с новой страницы, предоставив компилятору возможность заполнить остаток текущей текстом, нужно воспользоваться командой `\afterpage`, реализованной в пакете afterpage (раздел 11.3.1). Однако, если поместить процедуру `longtable` в аргумент команды `\afterpage`, это может вызвать нехватку памяти. Поэтому рекомендуется всю процедуру переписать в отдельный файл (назовем его `ltfile.tex`), а затем прочитать его командой `\input`. Последнюю следует поместить в аргумент команды `\afterpage` вслед за командой `\clearpage`:

```
\afterpage{\clearpage\input{ltfile}}
```

Чтобы запретить допечатку текста вслед за длинной таблицей на её последней странице, надо добавить ещё одну команду `\clearpage`:

```
\afterpage{\clearpage\input{ltfile}\clearpage}
```

12.6. Раскраска таблиц

Изменение цвета ячеек, столбцов или строк таблицы оказывается не очень простой задачей. Частично её решает пакет color (раздел 10.7), обеспечивая единый интерфейс с разнообразными выходными устройствами, но полное решение даёт пакет colortbl Дэвида Карлайла (Carlisle, David), специально написанный для раскраски таблиц.

Пакет colortbl автоматически загружает стандартные пакеты color, array и может использоваться совместно с пакетами longtable, dcolumn, hhline и т. д. Для изменения цвета фона отельной колонки или ячейки он вводит команду

`\columncolor[model]{clr}[l-hang][r-hang]` (colortbl)

Её можно использовать в качестве спецификатора столбцов `>{decl}`, внутри преамбулы таблицы cols или в описании столбцов команды `\multicolumn`. Эта команда задаёт цвет фона данного столбца или ячейки. Аргументы model (модель цвета) и clr (цвет) имеют тот же смысл, что и в командах, определённых в пакете color. Необязательные аргументы l-hang и r-hang определяют, насколько далеко влево и вправо от столбца простирается цветной фон. По умолчанию они равны величине `\tabcolsep` (в случае процедуры `tabular`) или `\arraycolsep` (в случае процедуры `array`), т. е. цвет полностью заполняет столбец.

Для начала раскрасим таблицу из примера на стр. 291[4]:

```
\setlength{\extrarowheight}{3pt}
\begin{tabular}[t]{|c|%
   >{\columncolor{yellow}\bfseries}l|%
   >{\columncolor{green}\itshape}c|}
 \hline \multicolumn{3}{|>{\columncolor{blue}}c|}
   {\color{white}{\bfseries Цвет}}       \\
 \hline A & B & C                        \\
 \hline 100 & 10 & 1                     \\
 \hline
\end{tabular}
```

Цвет		
A	**B**	*C*
100	**10**	*1*

Чтобы изменить цвет фона целой строки, предусмотрена команда

`\rowcolor[model]{clr}[l-hang][r-hang]` (colortbl)

Её следует помещать в начало строки. Особенное эффектно выглядит таблица с чередованием цвета строк:

```
\begin{tabular}[t]{|c|>{\bfseries}l|>{\itshape}c|}
  \rowcolor[gray]{0.7}  \hline A   & B  & C \\
                        \hline 100 & 10 & 1 \\
  \rowcolor[gray]{0.85} \hline 200 & 20 & 2 \\
                        \hline 300 & 30 & 3 \\
  \rowcolor[gray]{0.85} \hline 400 & 40 & 3 \\
                        \hline
\end{tabular}
```

A	**B**	*C*
100	**10**	*1*
200	**20**	*2*
300	**30**	*3*
400	**40**	*3*

На пересечении цветных столбцов и строк преимущество имеют установки команды окрашивания строки `\rowcolor`, но их действие ограничено текущей строкой.

Цвета рамки и разделительных линий между ячейками таблицы, созданные командами `\hline`, `\cline`, `\vline` или символом `|` в преамбуле таблицы cols, изменяет декларация

`\arrayrulecolor[model]{clr}` (colortbl)

[4] Реальные цвета в таблице заменены оттенками серого.

Она действует на все последующие линии и может располагаться вне таблицы, в начале строки или внутри спецификации >{decl} в преамбуле таблицы cols. Её влияние не ограничивается текущей областью действия деклараций, но не отменяет спецификации, заданные в cols.

```
\setlength{\arrayrulewidth}{2pt}
\arrayrulecolor[gray]{0.7}
\begin{tabular}[t]{|c|>{\bfseries}l|>{\itshape}c|}
    \arrayrulecolor[gray]{0.3} \hline A & B & C    \\
    \arrayrulecolor[gray]{0.5} \hline 100 & 10 & 1 \\
                               \hline 200 & 20 & 2 \\
    \arrayrulecolor[gray]{0.3} \hline
\end{tabular}
```

A	**B**	*C*
100	**10**	*1*
200	**20**	*2*

Промежуток между двойными линиями, созданными двумя символами `||` в преамбуле таблицы cols или двумя командами `\hline` в теле таблицы, можно окрасить с помощью декларации

`\doublerulecolor[model]{clr}` (colortbl)

Если окраска линий производится одновременно с окрашиванием фона в ячейках, частичные линии, созданные командой `\cline`, приобретают цвет фона, становясь невидимыми. Чтобы достичь желаемого эффекта в этом случае, следует загрузить пакет hhline.

Господи, благослови того пользователя,
который ничего не ждёт взамен разочарования.
Устав Франклина

Глава 13

Библиография и цитирование литературы

Публикации научно-технического направления обычно содержат список цитируемой литературы. Современный метод цитирования литературы удачно вписывается в схему организации перекрёстных ссылок, реализованную в LaTeX'е (раздел 3.7). Список литературы обычно состоит из пронумерованных записей, в каждой из которых указаны авторы, название, место и время публикации одного, реже двух или более источников, таких как книга или статья в журнале. Чтобы отослать Читателя к нужному источнику, в тексте печатного документа просто указывают его номер в списке литературы. Иногда вместо номера пишут фамилии авторов и год издания.

LaTeX имеет специализированную процедуру `thebibliography` для составления списка литературы. Она очень похожа на процедуру `enumerate`, которая печатает список пронумерованных записей. В отличие от последней, процедура `thebibliography` дополнительно печатает заголовок списка — слово «Литература» или что-то в этом роде, а каждая запись начинается с команды `\bibitem`, совмещающей функции команд `\item` и `\label`. Отличается также команда, печатающая номер ссылки на запись в списке литературы, а именно: вместо `\ref` нужно использовать команду `\cite`.

Список литературы `thebibliography` можно составлять заново для каждого печатного документа либо поручить это занятие программе BibTeX. Она компилирует список цитируемой литературы, извлекая информацию из заранее составленной библиографической базы данных. BibTeX поставляется вместе с LaTeX'ом. Библиографическую базу данных Читатель сможет составить самостоятельно, а затем расширять её по мере необходимости.

13.1. Процедура `thebibliography`

Список цитируемой литературы печатает процедура

```
\begin{thebibliography}{wlab}  ...  \end{thebibliography}
```

Обычно она помещается в конце печатного документа. Заголовок списка литературы зависит от класса документа; его легко изменить, переопределив команду `\bibname` или `\refname` (см. ниже).

Аргумент *wlab* процедуры `thebibliography` служит для задания левой границы списка литературы. LaTeX определяет ширину текста, содержащегося в *wlab*, как если бы тот был напечатан, и сдвигает левую границу текста в списке на соответствующее расстояние. Номера (или метки) записей в списке печатаются в образовавшейся свободной колонке слева от границы текста. Поэтому рекомендуется, чтобы ширина текста в *wlab* была равна самому широкому номеру (или самой широкой метке) записей в списке литературы.

Каждая запись в теле процедуры `thebibliography` должна начинаться с команды

```
\bibitem[label]{key}
```

Она генерирует запись, помеченную меткой *label*. Если опция *label* пропущена, запись получает очередной порядковый номер в списке литературы. Этот номер хранится в счётчике `enumiv`. Обязательный аргумент *key* является ключом, по которому команда

⚠
```
\cite[text]{key}
```

печатает номер или метку *label* соответствующей записи.

```
В книге \cite{Ivanov:86} приведено
полное описание процесса.
        ...
\begin{thebibliography}{99}
...
\bibitem{Ivanov:86}
\emph{А.\,Б.\ Иванов}. Огурцы на
подоконнике. Урюпинск, 1986.
\end{thebibliography}
```

В книге [16] приведено полное описание процесса.

...

Литература

...

[16] *А. Б. Иванов.* Огурцы на подоконнике. Урюпинск, 1986.

В этом примере «99» в аргументе процедуры `thebibliography` имеет такую же ширину, как и двузначные номера записей в списке литературы. Процедура `thebibliography` печатает записи в порядке их расположения в её теле в исходном тексте.

В качестве ключа *key* может выступать любая последовательность букв, цифр и знаков пунктуации, исключая запятую. Прописные и строчные буквы рассматриваются как разные символы. Русские буквы нельзя использовать в ключе *key*, но если очень хочется, то нужно загрузить пакет citehack из коллекции T2 Владимира Воловича, причём после пакета babel. Многие авторы используют гарвардский стиль в ключах *key*. В этом случае сначала указывают имя автора, а затем год издания. В гарвардском стиле составлен ключ `Ivanov:86` в приведённом примере. Однако каких-либо жёстких ограничений на способ составления ключей не существует.

Одной командой `\cite` можно сослаться на несколько записей в списке литературы. Ключи записей в аргументе `\cite` надо перечислять через запятую. Необязательный аргумент *text* в команде `\cite` служит для уточнения ссылки на источник. Чаще всего *text* указывает на номер страницы или параграфа.

```
Смотри \cite{Ivanov:2001, Petrov:2002},
а также \cite[\S5]{Sidorov:2003}.
```

Смотри [2, 3], а также [9, §5].

При наличии необязательного аргумента в команде `\bibitem` именно он используется в качестве метки записи. Номер записи не печатается, а счётчик `enumiv` не увеличивается.

```
...в статье \cite{t21}.
          ...
\begin{thebibliography}{W:99}
...
\bibitem[X:93]{t21}
\emph{Р.\,И. Хамлов}. Устройство
РИХ для прокладки туннелей.
Наука, 1993.
\end{thebibliography}
```

...в статье [X:93].

...

Литература

...

[X:93] *Р. И. Хамлов.* Устройство РИХ для прокладки туннелей. Наука, 1993.

В этом примере «X:93» является меткой. Необязательный аргумент *label* команды `\bibitem` является подвижным, поэтому хрупкие команды в ней следует защищать, предваряя их командой `\protect`.

Формат списка литературы, как и формат ссылок, печатаемых командой `\cite`, определяется классом печатного документа. Издательства и редакции журналов распространяют специализированные классы, которые устанавливают формат ссылок сообразно редакционным требованиям, если эти требования отличаются от принятых в стандартных классах. Разработаны также пакеты, которые печатают номера ссылок не в квадратных скобках, а в виде верхних индексов. Иногда эти пакеты изменяют команду `\cite` так, что она автоматически форматирует список из трёх или более ссылок, перечисленных в её аргументе через запятую, в виде диапазона. К наиболее популярным пакетам такого рода следует отнести cite и overcite Дональда Арсено (Arseneau, Donald), а также natbib Пэтрика Дейли (Daly, Patrick). Ссылка вида [4, 2, 8, 3] после загрузки пакета cite будет преобразована к виду [2–4, 8], а пакет natbib позволяет даже управлять порядком ссылок при перечислении. По умолчанию этот пакет оставит [4, 2, 8, 3] в неизменном виде, поскольку ссылки 2, 3, 4 перечислены не по порядку, однако если пакет будет загружен с опцией `sort&compress`, то ссылка будет напечатана в виде [2–4, 8]. Пакет overcite принудительно печатает ссылки в виде верхних индексов, и [4, 2, 8, 3] превратится в $^{2\text{–}4,\,8}$. Указанные пакеты также вводят дополнительные команды, которые позволяют варьировать формат ссылок, но мы не будем на этом останавливаться.

Дополнительные возможности форматирования списка литературы предоставляет использование команды

```
\newblock
```

в теле процедуры `thebibliography`. Её вставляют между логически различимыми частями описания каждой записи в списке литературы. Например, BIBTEX ставит `\newblock` после списка авторов, названия журнала, названия издательства и т. д. Стандартные классы определяют команду `\newblock` так, что обычно она просто вставляет дополнительный горизонтальный пробел:

```
\bibitem[X:93]{t21}
\emph{P.\,И. Хамлов}.
\newblock Устройство РИХ для
прокладки туннелей.
\newblock Наука, 1993.
```

...

[X:93] *Р. И. Хамлов.* Устройство РИХ для прокладки туннелей. Наука, 1993.

При наличии опции `openbib` в `\documentclass` каждый логический блок начинается с новой строки, а последующие строки в блоке сдвигаются вправо на дополнительное расстояние `\bibindent`:

```
\bibitem[X:93]{t21}
\emph{P.\,И. Хамлов}.
\newblock Устройство РИХ для
прокладки туннелей.
\newblock Наука, 1993.
```

[X:93] *Р. И. Хамлов.*
Устройство РИХ для прокладки туннелей.
Наука, 1993.

Для организации цитирования литературы, как и для организации любых других видов перекрёстного цитирования, LATEX использует информацию, которую он записал в служебный файл *jobname*`.aux` при предыдущей обработке входного файла. Поэтому после внесения изменений в список литературы входной файл должен быть обработан дважды.

13.1.1. Параметры настройки

`\bibindent` — нерастяжимая длина, используемая для форматирования списка литературы при наличии опции `openbib` в `\documentclass`.

`\refname` — логос заголовка списка литературы в классе `article`.

`\bibname` — логос заголовка списка литературы в классах `book` и `report`.

Логосы переопределяются с помощью `\renewcommand`. Например:

```
\renewcommand{\bibname}{Библиография}
```

Длина `\bibindent` изменяется командами `\setlength` и `\addtolength`.

13.2. BibTeX

BibTeX — это отдельная программа, которая собирает из библиографической базы данных список цитируемой литературы для печатного документа и сортирует его либо по очерёдности упоминания ссылок в тексте, либо по именам авторов. Программу написал Орен Паташник (Patashnik, Oren) [18]. Позднее Нил Кемпсон (Kempson, Niel) и Алехандро Агуилар-Сьерра (Aguilar-Sierra, Alejandro) разработали международную версию программы `bibtex8`, которая легко адаптируется ко всем европейским языкам. Настройка на тот или иной язык производится при запуске программы, когда она считывает текстовый файл, где указано соответствие прописных и строчных букв друг другу, а также их порядок по алфавиту. Например, для нашей книги сборку списка литературы производит команда

```
bibtex8 --csfile cp1251.csf manual.aux
```

где `manual.aux` — служебный файл, созданный компилятором, а `cp1251.csf` — тот самый файл настройки с русскими буквами, упорядоченными по алфавиту; как правило, он имеет расширение `csf`. Имя файла отражает используемую нами кодировку `cp1251` исходного текста во входном файле, которая соответствует русской версии Windows. В комплекте `bibtex8` имеется файл настройки `cp866rus.csf`, предназначенный для русской версии MS DOS. Файл с любой другой русской кодировкой можно сделать из `cp866rus.csf`, просто изменив в нём кодировку, но мы нашли нужный нам `cp1251.csf` в комплекте библиографических стилей, разработанных Максимом Поляковым специально для печати библиографического указателя на русском и украинском языках.

При использовании библиографической базы данных список литературы вместо процедуры `thebibliography` печатает команда

```
\bibliography{bib-files}
```

где bib-files — список файлов, содержащих библиографические данные. Эти файлы должны иметь расширение `bib`. Список может состоять из одного или нескольких файлов; в последнем случае имена файлов должны быть разделены запятыми. Например, команда

```
\bibliography{mhd,wall}
```

определяет, что список литературы будет выделен из библиографических данных, содержащихся в файлах `mhd.bib` и `wall.bib`. Подобные файлы состоят из записей, формируемых по определённым правилам, которые будут описаны в разделе 13.3. Один и тот же файл данных может быть использован для формирования списка литературы в различных документах.

При использовании процедуры `thebibliography` оформление списка цитируемой литературы почти целиком лежит на наборщике текста. Напротив, команда `\bibliography` в совокупности с BibTeX'ом позволяет варьировать формат списка литературы путём изменения *библиографического стиля*. Библиографический стиль выбирает декларация

`\bibliographystyle{`*bib-style*`}`

Она должна предшествовать команде `\bibliography`, так как BibTeX не знает, какой стиль использовать по умолчанию[1]. Для начала можно выбрать стиль `plain`:

```
\bibliographystyle{plain}
\bibliography{mhd,wall}
```

Он относится к числу четырёх стандартных стилей.

`plain` — открытый стиль. Библиография сортируется в алфавитном порядке по фамилиям авторов, затем по годам, затем по названиям публикаций. Каждая запись помечается порядковым номером. При ссылке на ту или иную публикацию её номер печатает команда `\cite`, используя ключ, которым публикация помечена в базе данных. Если имеются библиографические ссылки на русском языке, для правильной сортировки программу `bibtex8` необходимо вызвать с ключом `--csfile`, как показано на стр. 312.

`unsrt` — несортирующий стиль. Аналогичен стилю `plain` за тем исключением, что записи в списке литературы печатаются в порядке их цитирования в тексте печатного документа. Этот стиль подходит для большинства научных журналов.

`alpha` — алфавитный стиль. Аналогичен стилю `plain`, но вместо нумерации публикаций они помечаются аббревиатурой, состоящей из фамилии автора и года издания (например, «Knu66»). Записи в списке литературы сортируются по меткам, затем по имени автора, затем по году издания, затем по названию.

`abbrv` — аббревиатурный стиль. Идентичен стилю `plain` за тем исключением, что вместо полных имен авторов, названий месяцев и журналов печатаются сокращения.

Специально для печати списка литературы на русском и украинском языках Максим Поляков разработал несколько стилей. Мы перечислим два из них, которые удовлетворяют ГОСТу 7.80-00 на оформление библиографического указателя, принятому в 2000 году:

`gost780s` — сортирующий стиль. Аналогичен стилю `plain`. Для правильной сортировки записей `bibtex8` необходимо запускать с ключом `--csfile`;

`gost780u` — несортирующий стиль. Аналогичен стилю `unsrt`.

Эти стили способны учитывать различия между библиографическими ссылками на разных языках. Например, перед именем редактора издания на русском языке будет вставлено слово «ред.» вместо английского слова «ed.», которое используют большинство стилей, не делающих различия между языками, в том числе все 4 стандартных библиографических стиля.

[1] Если *bib-style* не выбран классом документа.

Любой библиографический стиль — это текстовый файл с расширением `bst`. Например, стилю `alpha` соответствует файл `alpha.bst`, который распространяется вместе с BibTeX'ом. С коллекцией пакетов AMS-LaTeX распространяются ещё два стиля: `amsalpha` и `amsplain`. Пакет `natbib` поставляется с собственным набором: `abbrvnat`, `plainnat` и `unsrtnat`. Класс `revtex4`, который мы рассмотрим в главе 15 в качестве примера нестандартных классов, также имеет собственные библиографические стили `apsrev` и `apsrmp`. Бессмысленно было бы описывать различия между всеми этими стилями. Проще испытать несколько стилей при работе с реальным документом и выбрать наиболее подходящий. Каждое издательство вправе выдвигать свои особые требования к оформлению списка литературы, поэтому перечисленные стили составляют лишь малую долю от множества существующих.

После того как команды `\bibliographystyle` и `\bibliography` расставлены в нужных местах входного файла, необходимо сделать следующие шаги.

1. Откомпилировать входной файл *jobname*`.tex`. Каждая команда `\cite` напечатает вместо номера ссылки два вопросительных знака, означающих, что ссылка не найдена. Как мы объясняли в разделе 3.7, это нормально. Одновременно в служебный файл *jobname*`.aux` будет записана информация о ключах всех ссылок; этот файл необходим BibTeX'у для создания файла *jobname*`.bbl`.

2. Обработать файл *jobname*`.aux` BibTeX'ом. Для этого нужно запустить программу `bibtex`, указав имя файла (можно без расширения) в качестве входного параметра. Обычно эта процедура автоматически выполняется из меню редактора документов LaTeX. Прочитав информацию из указанных в `\bibliography` файлов библиографической базы данных (с расширением `bib`), BibTeX создаст файл *jobname*`.bbl`, куда запишет текст, оформленный в виде процедуры `thebibliography`.

3. Вторично обработать входной файл LaTeX'ом. Команда `\bibliography` вставит список литературы в печатный документ из файла *jobname*`.bbl`.

4. В третий раз обработать входной файл LaTeX'ом. Будут вставлены ссылки на литературу.

Теперь печатный документ готов. Такая четырёхшаговая процедура имеет тот недостаток, что добавление или удаление ссылки может потребовать повторного запуска BibTeX'а. Преимущество же состоит в том, что файл *jobname*`.bbl` можно при желании подредактировать, пользуясь любым текстовым редактором, если составленный BibTeX'ом список литературы не удовлетворит Читателя.

Иногда необходимо включить в список литературы ту или иную публикацию, не делая на неё ссылок в тексте печатного документа. Эту задачу решает команда

⚠ `\nocite[`*text*`]{`*key*`}`

Она, как и \cite, заносит ссылку (одну или несколько) в список литературы, но сама ничего не печатает. Она может появиться в любом месте после команды \begin{document}. Одной командой \nocite{*} можно включить в список литературы все записи из базы данных без явного их перечисления в командах \cite или \nocite.

13.3. Библиографическая база данных

Нет смысла создавать базу данных для одного печатного документа, так как на это уйдёт больше времени, чем на составление списка литературы при помощи процедуры thebibliography. Но при систематическом использовании библиографическая база данных станет мощным подспорьем в работе, как представляющая самостоятельную ценность в качестве источника информации.

Любой файл библиографической базы данных с расширением bib содержит последовательность записей. Каждая запись начинается символом @, за которым следует признак типа записи. Например, следующая запись имеет тип book:

```
@BOOK{knydson,
      AUTHOR    = "Ларсен Кнудсон",
      TITLE     = "Альманах Путешествий",
      PUBLISHER = {Наука},
      ADDRESS   = {Новосибирск},
      LANGUAGE  = "russian"
}
```

Перечень записей допустимых типов приводится в разделе 13.4. Аббревиатура knydson есть ключ *key* для команды \cite{*key*}, по которому она ссылается на запись. Данная запись имеет пять полей: author (автор), title (название), publisher (издатель), address (адрес), а также language (язык). Поле состоит из знака =, возможно, окружённого пробелами слева и справа, и текста. Текст состоит из строки символов, окружённых с обеих сторон двойными кавычками (") или фигурными скобками ({ и }). В строке символов не должно быть непарных фигурных скобок, причём, в противоположность LaTeX'у, \{ и \} также учитываются при подсчёте фигурных скобок. Ключ и все поля разделяются между собой запятыми. Вслед за последним полем она не ставится. Самые внешние фигурные скобки в записи могут быть заменены круглыми. Так же, как и в LaTeX'е, невидимый символ признака конца строки рассматривается как обычный пробел, а сто пробелов подряд эквивалентны одному. В отличие от LaTeX'а, BibTeX не делает различия между прописными и строчными буквами в служебных словах. Поэтому не будет ошибкой написать AUTHOR вместо author. Двойные кавычки вокруг текста, состоящего из цифр, могут быть опущены, поэтому поля Volume = "17" и Volume = 17 эквивалентны.

13.3.1. Текст поля

Текст поля заключается в фигурные скобки или в двойные кавычки ("). Говорят, что текст заключён в фигурные скобки, если он находится внутри пары скобок { и }, отличной от пары скобок, окружающей всю запись целиком.

Имя

Текст поля `author` (автор) или `editor` (редактор) представляет собой список фамилий авторов и издателей соответственно. Библиографический стиль определяет, будут ли напечатаны сначала имя и отчество (инициалы) автора (издателя), а затем его фамилия, или наоборот.

Записи в файле базы данных просто содержат информацию, что именно является именем, отчеством и фамилией. Большинство фамилий и имён могут быть введены в базу данных очевидным способом, с запятой или без неё, как показано в следующем примере:

```
AUTHOR = "Иван Петрович Сидоров"     AUTHOR = "Сидоров, Иван Петрович"
AUTHOR = "Ludwig von Beethoven"      AUTHOR = "von Beethoven, Ludwig"
```

Однако для людей, имеющих множественные фамилии, следует использовать только форму с запятой. Например, Per Brinch Hansen, чьё имя Per, а фамилия Brinch Hansen, должен быть записан так: `"Brinch Hansen, Per"`. Если напечатать `"Per Brinch Hansen"`, то алгоритм, используемый BibTeX'ом, примет Brinch за среднее имя, то есть отчество. Имена `von Beethoven` или `de la Madrid` проблем не вызывают, потому что `von` и `de la` написаны со строчной буквы.

Имя и фамилию автора целесообразно набирать в точности так, как они фигурируют в цитируемой публикации, даже если в другой работе имя того же автора выглядит немного иначе:

```
AUTHOR = "Donald E. Knuth"           AUTHOR = "D. E. Knuth"
```

Если есть уверенность, что два разных представления имени принадлежат одному лицу, можно в обоих случаях использовать одну и ту же форму, а именно ту, которую предпочитает сам автор. В последнем примере это `"Donald E. Knuth"`. Тем не менее во втором случае необходимо указать, что в оригинале имя автора записано по-другому:

```
AUTHOR = "Donald E. Knuth"           AUTHOR = "D[onald] E. Knuth"
```

При подобных предосторожностях список литературы будет правильно отсортирован во всех мыслимых и немыслимых ситуациях, а BibTeX самостоятельно определит сообразно выбранному библиографическому стилю, как следует сокращать имена.

BibTeX рассматривает текст в фигурных скобках как нечто целое, так что фигурные скобки следует использовать для устранения возможных разночтений. Например, скобки в

```
"{Barnes and Noble, Inc}"
```

исключают интерпретацию `Inc` в качестве имени. BibTeX рассматривает `von Beethoven, Ludwig` и `{von Beethoven}, Ludwig` как два различных имени. В первом случае `Beethoven` означает фамилию, а `von` — частица. Во втором случае `von Beethoven` — это фамилия. Библиографический стиль, возможно, не сделает различий между этими двумя именами, но при расстановке их в алфавитном порядке отведёт им разные позиции.

Слово `Junior` (младший) вызывает особые проблемы. Большинство людей с Jr. в своём имени предваряют его запятой. В этом случае имя должно быть введено так:

```
"Ford, Jr., Henry"
```

Однако некоторые люди не используют запятую, они рассматривают Jr. как часть фамилии:

```
"{Berk Jr.}, Herbert L."
```

В самом общем случае полное имя может состоять из фамилии $Last$, имени и отчества $First$, указание старшинства Jr и частицы von, но только $Last$ является обязательным. Частицей von считается слово (или слова), начинающиеся со строчной буквы. Остальные части в затруднительных случаях нужно писать через запятую в следующем порядке: $von\ First$, Jr, $Last$ или $von\ First$, $Last$.

Некоторые особенности имеет применение в BibTeX'е символов с диакритическими знаками. Например, если имеются два поля

```
AUTHOR = "Kurt G{\"o}del",
YEAR   = 1931,
```

и используется стиль `alpha`, тогда BibTeX сконструирует для этой записи метку [Göd31], как и следовало ожидать. В общем случае, чтобы обеспечить правильную обработку символа с диакритическим знаком, необходимо его целиком окружить фигурными скобками; в данном случае верным решением будет `{\"o}` или `{\"{o}}`. Более того, эти фигурные скобки не должны быть окружены другими фигурными скобками (кроме самых внешних, которые содержат всё поле), а первым символом внутри скобок должен быть обратный слеш. Поэтому ни `{G{\"{o}}del}`, ни `{G{\"o}del}` не сработают ожидаемым образом при сортировке списка литературы.

Символ с диакритическим знаком является частным случаем «спецсимвола». Таковым для BibTeX'а считается весь текст, начиная с левой фигурной скобки, за которой сразу же следует обратный слеш \, и заканчивая парной ей правой фигурной скобкой. Например,

```
AUTHOR = "van R{\i\j}den"
```

содержит спецсимвол `{\i\j}`. BibTeX не обрабатывает команды LaTeX'а внутри спецсимвола. Например, при использовании стиля, переводящего все заглавия в нижний регистр, т. е. печатающего их строчными буквами, BibTeX преобразует запись

```
THE {\TeX BOOK}
```

в `The \TeX book`.

При наличии нескольких авторов их фамилии разделяются словом `and`. Статья, написанная Alpher'ом, Bethe и Gamow'ым, имеет следующее поле авторов:

```
AUTHOR = "Alpher, Ralf and Bethe, Hans and Gamow, George"
```

Список авторов становится более удобочитаемым, если имя каждого автора выделено скобками:

```
AUTHOR = "{Alpher, Ralf} and {Bethe, Hans} and {Gamow, George}"
```

Слово `and` разделяет авторов только в том случае, если оно не окружено фигурными скобками. Следовательно, если слово `and` является частью имени, оно должно быть окружено фигурными скобками, как в примере с `{Barnes and Noble, Inc}`, приведённом выше.

Название

Библиографический стиль определяет, будут или нет все слова в названии публикации печататься прописными буквами. В названии книги обычно все буквы прописные, а в названии статьи — только некоторые. Если в названии слова должны начинаться с прописной буквы, то и набирать их надо соответствующим образом:

```
TITLE = "The Agony and the Ecstasy"
```

В названиях иностранных изданий с прописной буквы принято начинать все слова, за исключением артиклей, союзов и предлогов. BibTeX заменяет все прописные буквы на соответствующие строчные, если того требует библиографический стиль. Те прописные буквы, которые не должны заменяться на строчные, необходимо заключить в фигурные скобки. Следующие два названия эквивалентны:

```
"Носороги и слоны {Африки}"        "Носороги и слоны {А}фрики"
```

Сокращение

Текст поля может быть сокращён до некоторой аббревиатуры. Аббревиатурой может быть строка символов, которая начинается буквой и не содержит пробела или любого из следующих десяти символов:

```
" # % ' ( ) , = { }
```

Аббревиатуру вводят на место текстового поля без фигурных скобок и кавычек. Если `жтф` есть аббревиатура строки

```
Журнал Технической Физики
```

то следующие поля эквивалентны:

```
JOURNAL = жтф
JOURNAL = "Журнал Технической Физики"
```

Некоторые аббревиатуры предопределены библиографическим стилем. Так, для названий месяцев на английском языке введены трёхбуквенные сокращения: `jan`, `feb`, `mar` и т. д. Библиографические стили обычно содержат аббревиатуры названий часто цитируемых журналов. В файле базы данных допускается определять дополнительные аббревиатуры с помощью команды `@string`. Например,

```
@STRING{жтф = "Журнал Технической Физики"}
```

определяет `жтф` как сокращение названия указанного журнала. Круглые скобки могут быть использованы вместо самых внешних фигурных скобок в аргументе команды `@string`, а фигурные скобки могут быть использованы вместо кавычек.

Команда `@string` может появляться до или между записями в файле базы данных. Однако она должна быть введена до первого использования соответствующей аббревиатуры, так что наилучшим местом для неё является начало файла. Команда `@string` в файле базы данных имеет старшинство перед определениями, вводимыми библиографическим стилем, так что она может переопределить сокращения, введённые стилем, как, например, сокращение `feb` для названия месяца февраль.

Склеивание строк

Несколько строк можно объединить в одну. Эта операция называется склеиванием и обозначается символом `#`. Например, если определена строка

```
@STRING{АП = "Альманах Путешествий"}
```

тогда легко произвести почти идентичные заголовки для нескольких различных записей:

```
@BOOK{АП-86,
      TITLE = 1986 # АП,
      ...
}
@BOOK{АП-87,
      TITLE = 1987 # АП,
      ...
}
```

и так далее. Можно соединять любое количество строк, ограниченное только общей длиной результирующей строки, определяемой конкретной реализацией BibTeX'а.

Язык

Стили `gost780s` и `gost780u` вносят лингвистические коррективы в оформление библиографической ссылки, если обнаруживают в соответствующей библиографической записи поля `language` или `booklanguage`. Стандартные стили игнорируют эти поля, а `gost780s` и `gost780u` распознают три возможных значения: `russian`, `ukrainian` и `english`, причём последнее используется по умолчанию. В большинстве случаев достаточно указать только `language`. Оба поля в одной записи имеет смысл применять для сборника статей на разных языках. В следующем примере дано описание статьи на русском языке в сборнике с заголовком на украинском языке:

```
@INPROCEEDINGS{Purich:1999,
    author = "Пурич, Штефан",
    title = "{Р}умынско-{У}краинское трансграничное сотрудничество:
      проблемы  и перспективы",
    language = "russian",
    booktitle = "Транскордонне співробітництво у поліетнічних регіонах
      {С}хідної та {П}івденно-{С}хідної {Є}вропи",
    editor = "Макара, Ю.",
    address = "Чернівці",
    publisher = "Золоті литаври",
    organization = "Буковинський політологічний центр",
    year = 1999,
    numpages = 196,
    booklanguage = "ukrainian"
}
```

13.3.2. Преамбула базы данных

Подобно преамбуле в корневом входном файле любой файл библиографической базы данных может иметь свою преамбулу. Она состоит из команды `@preamble`. Её синтаксис подобен команде `@string`, но не содержит имени строки и знака равенства:

```
@PREAMBLE{ "\newcommand{\nosort}[1]{} # ... " }
```

В данном примере преамбула содержит определение новой команды `\nosort` (и, возможно, других, перечисленных через знак `#`). Команда `\nosort{`*arg*`}` имеет один аргумент, но в конечном итоге прячет его (глава 7). Такую команду можно использовать для изменения режима сортировки записей в списке литературы. Например, если составить для двух изданий одной и той же книги поле `year` следующим образом:

```
YEAR = "{\nosort{a}}1983"
. . .
YEAR = "{\nosort{б}}1971"
```

то BibTeX (при использовании стиля `plain`) первой в список литературы поместит ссылку на более позднее издание, потому что в алфавите буква `а` идёт ранее буквы `б`, а BibTeX будет считать, будто команда `\nosort` производит какой-нибудь особый диакритический знак.

13.3.3. Перекрёстное цитирование

В дополнение к обычной схеме перекрёстного цитирования BibTeX располагает своей собственной. В следующем примере первая запись ссылается на вторую:

```
@INPROCEEDINGS{no-gnats,
      CROSSREF = "gg-proc",
      AUTHOR   = "Rocky Gneisser",
      TITLE    = "No Gnats Are Taken for Granite",
      PAGES    = "133-139"}
. . .
@PROCEEDINGS{gg-proc,
      EDITOR    = "Gerald Ford and Jimmy Carter",
      TITLE     = "The Gnats and Gnus 1988 Proceedings",
      BOOKTITLE = "The Gnats and Gnus 1988 Proceedings"}
```

Поле `crossref` в записи `no-gnats` указывает, что эта запись должна наследовать любые пропущенные поля из записи `gg-proc`. В данном случае наследуются `editor` и `booktitle`.

Более того, BibTeX автоматически вставит запись `gg-proc` в список литературы, если на неё не менее двух раз ссылаются другие записи, даже если запись `gg-proc` не была процитирована командами `\cite` или `\nocite`.

Чтобы гарантировать правильность работы этой схемы перекрёстного цитирования, запись, на которую ссылаются другие записи, должна появляться в базе данных после этих записей.

Обычная схема перекрёстного цитирования также действует. Например, можно сделать ссылку на какую-нибудь публикацию в примечаниях к другой ссылке:

```
NOTE = "Джонсон \cite{jones:86} уточнил этот результат"
```

13.4. Записи в базе данных

Первое, что нужно сделать при добавлении в базу данных новой записи,— это решить, к какому типу следует её отнести. На этот счёт не существует жёстких правил, но BibTeX обеспечивает достаточное число типов записей, чтобы удовлетворить самым разнообразным потребностям.

Ссылки на разные публикации содержат различную информацию. Например, ссылка на журнальную статью может включать том и номер журнала, которые, как правило, отсутствуют при ссылке на книгу. Следовательно, записи в базе данных для различных типов имеют различный набор полей. Для каждого типа записей поля делятся на три класса.

`required` — обязательное поле. Пропуск поля вызывает сообщение об ошибке и может плохо отразиться на формировании записи в списке литературы. Если требуемая информация не является осмысленной, то выбран неправильный тип записи. Однако, если требуемая информация содержательна, но уже включена в другие поля, можно проигнорировать предупреждение об ошибке.

`optional` — необязательное поле. Информация поля будет использована, если она присутствует, но она может быть опущена без ущерба для форматирования записи в списке литературы. Ссылка на публикацию должна содержать любую информацию, которая может помочь отыскать её. Поэтому полезно заполнять необязательные поля, если они предусмотрены типом записи.

`ignored` — игнорируемое поле. BIBTEX игнорирует любое поле, которое для выбранного типа записи не входит в предыдущие два класса. Поэтому по желанию можно добавлять новые поля в записи. Хорошей идеей является включение всей полезной информации о ссылке в соответствующую запись в файле базы данных, даже если часть её никогда не появится при составлении списка литературы. Например, если нужно хранить аннотацию публикации в базе данных, можно добавить поле `abstract`. Библиографическая база данных может быть хорошим местом для комментирования прочитанных статей и книг тем более, что можно создать библиографический стиль для печати аннотаций.

Ниже перечислены все типы записей со всеми обязательными и необязательными полями, которые используются стандартными библиографическими стилями. Назначение каждого из полей объясняется в следующем разделе. Выбранный библиографический стиль может игнорировать некоторые необязательные поля при составлении списка литературы. Напомним, что признаку типа записи в файле базы данных должен предшествовать знак `@`.

`article` — статья в журнале. Обязательные поля: `author`, `title`, `journal`, `year`. Необязательные поля: `volume`, `number`, `pages`, `month`, `note`.

`book` — книга с указанием издательства. Обязательные поля: `author` или `editor`, `title`, `publisher`, `year`. Необязательные поля: `volume` или `number`, `series`, `address`, `edition`, `month`, `note`.

`booklet` — отпечатанное и сброшюрованное издание без названия издательства или имени спонсора. Обязательные поля: `title`. Необязательные поля: `author`, `howpublished`, `address`, `month`, `year`, `note`.

`conference` — статья в трудах конференции. Обязательные поля: `author`, `title`, `booktitle`, `year`. Необязательные поля: `editor`, `pages`, `organization`, `publisher`, `address`, `month`, `note`.

`inbook` — часть книги, которая может быть главой или рядом страниц. Обязательные поля: `author` или `editor`, `title`, `chapter` и/или `pages`, `publisher`, `year`. Необязательные поля: `volume` или `number`, `series`, `type`, `address`, `edition`, `month`, `note`.

`incollection` — часть книги со своим заглавием. Обязательные поля: `author`, `title`, `booktitle`, `publisher`, `year`. Необязательные поля: `editor`, `volume` или `number`, `series`, `type`, `chapter`, `pages`, `address`, `edition`, `month`, `note`.

`inproceedings` — то же, что и `conference`.

`manual` — техническая документация. Обязательные поля: `title`. Необязательные поля: `author`, `organization`, `address`, `edition`, `month`, `year`, `note`.

`mastersthesis` — дипломная работа. Обязательные поля: `author`, `title`, `school`, `year`. Необязательные поля: `address`, `month`, `note`.

`misc` Используйте этот тип, когда нет ничего более подходящего. Обязательных полей нет. Необязательные поля: `author`, `title`, `howpublished`, `month`, `year`, `note`.

`phdthesis` — диссертация. Обязательные поля: `author`, `title`, `school`, `year`. Необязательные поля: `address`, `month`, `note`.

`proceedings` — труды конференции. Обязательные поля: `title`, `year`. Необязательные поля: `editor`, `publisher`, `organization`, `address`, `month`, `note`.

`techreport` — доклады, опубликованные школами или институтами, обычно пронумерованные внутри серии. Обязательные поля: `author`, `title`, `institution`, `year`. Необязательные поля: `type`, `number`, `address`, `month`, `note`.

unpublished — неопубликованный документ с автором и заглавием. Обязательные поля: author, title, note. Необязательные поля: month, year.

Запись любого типа может иметь необязательное поле key (ключ), которое в некоторых стилях определяет порядок расположения ссылок по алфавиту или формирования меток командой \bibitem. Необходимо включить поле key в любую запись, у которой поля author и editor опущены. Не путайте поле key с ключом *key*, который появляется в аргументе команды \cite и в начале каждой записи после указания её типа.

13.5. Поля записей в базе данных

Ниже приводится описание всех полей, распознаваемых стандартными библиографическими стилями. Записи могут также содержать другие поля, игнорируемые этими стилями.

address — адрес издателя. Для крупных издательств достаточно указать только город. Для малых издательств желательно привести полный адрес.

annote — аннотация. Стандартными стилями не используется, но может быть использована другими стилями, предназначенными для создания аннотированных библиографических указателей.

author — фамилия автора. Несколько фамилий разделяются словом and.

booklanguage — язык цитируемой книги. Распознается стилями gost780u и gost780s. Допустимы значения russian, ukrainian и english (используется по умолчанию). Применяется для книги, содержащей статьи на нескольких языках.

booktitle — заголовок цитируемой книги. Для записи типа book вместо booktitle следует использовать title. Об особенностях цитирования названий книг см. раздел 13.3.1.

chapter — номер главы, раздела, параграфа и т. д.

crossref — ключ, указывающий на запись в библиографической базе данных, на которую имеются перекрёстные ссылки из других записей (раздел 13.3.3).

edition — номер издания книги, например второе.

editor — фамилия редактора. Если ещё имеется поле author, поле editor указывает редактора книги или сборника, в котором содержится цитируемая публикация.

howpublished — резервное поле для публикаций, изданных необычным путём.

institution — институт, поддерживающий работу, или спонсорская организация.

journal — название журнала. В библиографической базе данных могут быть определены сокращения для названий некоторых журналов.

language — язык цитируемого источника. Распознается стилями gost780u и gost780s. Допустимы значения russian, ukrainian и english (используется по умолчанию).

key — ключ для сортировки в алфавитном порядке и создания меток в тех случаях, когда поля author и editor опущены. Поле key не надо путать с ключом *key* в аргументах команд \cite, \nocite и в начале каждой записи в базе данных.

month — месяц опубликования работы или, если она не была опубликована, время написания. Следует использовать стандартные сокращения названий месяцев.

note — любая дополнительная информация, которая может помочь Читателю. Буквы в первом слове переводятся в прописные.

`number` — номер журнала, технического доклада или выпуска в серийных изданиях. Номер журнала обычно состоит из номера тома и номера выпуска в томе. Организация, которая выпускает технический отчёт (препринт), обычно присваивает ему номер по своим правилам.

`organization` — спонсор конференции или издания.

`pages` — номера страниц (одной и более) или диапазон страниц. Например, `24`, или `24, 31`, или `37--66`, или `24, 31, 37--66`. Стандартные стили трансформируют одинарный дефис (`37-66`) в двойной (`37--66`), как того требуют правила LaTeX'а.

`publisher` — название издательства.

`school` — название учебного заведения, где работа была написана.

`series` — название серии книг. Когда цитируется книга, поле `title` даёт её название, а необязательное поле `series` даёт название серии, в которой книга опубликована.

`title` — название публикации.

`type` — вид технического доклада, например «для служебного пользования».

`volume` — том журнала или том книги в многотомном издании.

`year` — год публикации или год написания для неопубликованной работы. Последние четыре символа, отличные от знаков препинания, должны быть цифрами, как «1993» или «(около 1834)».

Мы по одной дороге ходим все.—
Так думал я.— Одно у нас начало,
Один конец.
Н. Рубцов. Философские стихи

Глава 14

Алфавитный указатель

Монографии и учебные пособия часто содержат разнообразные указатели. По подбору терминов различают предметные, тематические, именные и иные указатели. По способу упорядочивания терминов выделяют алфавитные, хронологические и систематические указатели. Как правило, указатель содержит список терминов с указанием страниц, где каждый термин встречается. Особый тип указателя составляет словарь терминов. Он обычно состоит из кратких пояснений каждого термина без ссылок на основной текст книги.

В этой главе мы рассмотрим составление алфавитного указателя. Уже из его названия ясно, что в таком указателе список терминов упорядочен по алфавиту. Различие указателей по подбору терминов мы обсудим лишь в минимальной степени, поскольку такие различия для компилятора совершенно несущественны. LaTeX предоставляет средства, позволяющие полностью автоматизировать составление алфавитного указателя и значительно облегчить подготовку словаря терминов. Текст алфавитного (равно как и любого другого) указателя печатает процедура `theindex`. Для словаря терминов нет специальной процедуры, так как вполне подойдёт процедура `description`, описанная в главе 4.

Гипотетически текст для `theindex` можно подготовить «вручную», хотя для этого пришлось бы вспомнить технологию докомпьютерной эры, когда термины для указателя выписывали на отдельных карточках и затем туда вписывали номера страниц, где каждый термин упоминается. Даже если однажды проделать эту титаническую работу, она пойдет в мусорную корзину перед следующим изданием книги, если будет добавлена хотя бы она страница. Поэтому в системе LaTeX сортировка терминов по алфавиту и отслеживание номеров страниц полностью автоматизированы. Наборщику текста нужно только пометить термины в исходном тексте командами `\index`.

Если затем загрузить пакет makeidx и вставить в преамбулу входного файла команду `\makeindex`, компилятор создаст *первичный указатель*, т. е. список всех помеченных терминов в порядке их упоминания в исходном тексте и сохранит его в отдельном файле. Далее этот список сортируют при помощи специальной программы. Поскольку любая реализация LaTeX'а содержит программу *MakeIndex*, чаще всего используют именно её. Она формирует упорядоченный по алфавиту список терминов в виде процедуры `theindex` и также сохраняет его в отдельном файле. Перед следующей компиляцией этот файл можно вставить в исходный

текст при помощи известной Читателю команды `\input` или более специализированной команды `\printindex`.

В разработке *MakeIndex* в разные годы принимали участие Пехонг Чжень (Chen, Pehong), Майкл Харрисон (Harrison, Michael) и Нил Кемпсон (Kempson, Niel). Консультировал работу Лесли Лампорт (Lamport, Leslie). *MakeIndex* можно настроить для сортировки словаря терминов, пометив нужные слова в исходном тексте при помощи команды `\glossary`. Дальнейшие действия идентичны подготовке алфавитного указателя за исключением самой последней стадии, так как аналога команды `\printindex` для словаря терминов нет. Предполагается, что отсортированный словарь терминов должен быть просто скопирован в исходный текст документа. На практике *MakeIndex* едва ли где-то применяют для сортировки словаря терминов, поскольку он, как правило, содержит не ссылки на номера страниц, а пояснения к терминам. Ясно, что бездушная программа не способна объяснить смысл термина. Тем не менее наличие двух команд: `\index` и `\glossary` — для выделения терминов несомненно полезно. В нашей книге мы отмечаем командой `\index` слова, которые должны попасть в предметный указатель, а команду `\glossary` используем для именного указателя, хотя её имя в буквальном переводе означает «толковый словарь».

14.1. Рецепт приготовления

Зафиксируем первые итоги нашего обсуждения, сосредоточившись для начала на составлении предметного указателя.

В качестве первого шага в исходном тексте следует пометить выбранные термины при помощи команды `\index`, пользуясь правилами, описанными в разделе 14.3. Если некоторые термины встречаются в тексте многократно, то, как говорится, возможны варианты. Например, можно пометить только первое упоминание каждого термина. Другая крайность — механически отметить каждое упоминание термина. Выбор варианта целиком определяется автором публикации. Обычно отмечают наиболее информативные упоминания каждого термина, но, например, в научных публикациях по истории часто фиксируется каждое упоминание события или имени исторического деятеля.

Чтобы пометить термин в исходном тексте документа в простейшем случае его достаточно продублировать в аргументе команды `\index`, которую размещают сразу вслед за термином. Команды `\index` безобидны, так как они попросту игнорируются, пока в преамбуле отсутствует декларация `\makeindex`. Поэтому метить термины можно на любой стадии редактирования исходного текста, отложив окончательную сборку алфавитного указателя до завершения основной части работы.

На завершающей стадии подготовки указателя в корневой входной файл (допустим, он называется *jobname*`.tex`) нужно внести следующие изменения:

1. Загрузить пакет makeidx[1] и вставить в преамбулу декларацию \makeindex. Она указывает, что LaTeX должен обновлять перечень помеченных терминов при очередной компиляции исходного текста. По традиции \makeindex помещают вслед за \usepackage{makeidx}:

```
\documentclass{book}
\usepackage[russian]{babel}
\usepackage{makeidx}
\makeindex
. . .
\begin{document}
. . .
```

2. Вставить команду \printindex там, где должен быть напечатан алфавитный указатель. Обычно её помещают ближе к концу документа.

3. Повторить компиляцию входного файла. В результате будет создан первичный указатель *jobname*.idx. Он состоит из *входов в указатель*, каждый из которых создан одной командой \index в исходном тексте.

4. Отсортировать первичный указатель программой *MakeIndex*. В результате будет создан файл *jobname*.ind, содержащий исходный текст алфавитного указателя в виде процедуры theindex. Обычно программа сортировки запускается из меню редактора исходных текстов LaTeX, так что Читателю не нужно беспокоиться, какой файл и чем обрабатывать.

При следующей компиляции исходного текста команда \printindex напечатает алфавитный указатель, взяв его текст из *jobname*.ind. Однако, если в указателе есть русские слова, он будет упорядочен как угодно, но только не по алфавиту. Такова «реальность, данная нам в ощущениях» авторами стандартной русификации LaTeX'а. Причиной всех проблем является принятое в 1999 году решение объявить русские буквы командами. LaTeX преобразует русские буквы в команды \cyr*cmd* и в таком виде записывает их во все служебные файлы, в том числе в файлы idx. Таким способом достигается совместимость документов с разметкой LaTeX, созданных на разных компьютерных платформах. Однако программа *MakeIndex* не обновлялась с 1993 года, когда русские буквы ещё были буквами[2]. Более современная программа xindy позволяет настраивать порядок сортировки, поэтому может правильно сортировать указатели гипотетически на любом существующем языке. Однако реализация xindy для новейших версий Windows на момент написания этой книги отсутствовала. В таких условиях приходится прибегать к кустарным методам. Они сводятся к принудительной перекодировке idx-файлов перед вызовом программы *MakeIndex*.

---

[1] В настоящее время бо́льшая часть функций пакета makeidx перенесена в ядро системы LaTeX. Пакет можно не загружать, если не используются команды \see, \seealso и \printindex.

[2] Прежний способ обработки русских букв без конвертации их в команды \cyr*cmd* можно смоделировать при помощи механизма TCX, см. раздел 16.5.2.

Разработчики «стандартной русификации» предлагают делать перекодировку с помощью скриптов `rumkidx`, которые вызывают программу `sed`, хорошо знакомую пользователям Unix. Установка `sed` в Windows возможна в составе библиотеки программ cygwin, но требует определённых усилий. На диске, прилагаемом к части тиража нашей книги, имеется скрипт `cyr2win.wsf`, который может работать с новейшими версиями Windows, начиная с Windows 98[3], без установки дополнительного программного обеспечения. Мы использовали `cyr2win.wsf` при подготовке алфавитного указателя для данного издания, обрабатывая все файлы с расширением `idx` перед вызовом *MakeIndex*:

```
cscript cyr2win.wsf jobname.idx
makeindx.exe jobname.idx
```

Паллиативное решение с принудительной перекодировкой первичного указателя решает только часть имеющихся проблем. С помощью *MakeIndex* невозможно осуществить «тонкую настройку» порядка сортировки путём введения дополнительных правил[4]. Например, нельзя поменять очерёдность прописных и строчных букв, которая в кодировке Windows различна для русских и латинских букв. Не исключено, что через какое-то время *MakeIndex* уступит своё место программе `xindy`. Она понимает все инструкции, предназначенные для *MakeIndex*, поэтому далее мы продолжим работу с *MakeIndex*, не полагаясь на дополнительные возможности `xindy`.

Сообщения об ошибках, генерируемые программой *MakeIndex*, описаны в приложении B.4. Анализ отсортированного алфавитного указателя может обнаружить дополнительные ошибки, не обнаруженные *MakeIndex*. Они могут быть исправлены за счёт изменения соответствующих команд `\index` в исходном тексте и повторной обработки документа. Отдельные исправления можно внести прямо в файл *jobname*`.ind`. Он содержит текст алфавитного указателя в виде процедуры `theindex`. Однако прямого редактирования текста алфавитного указателя желательно избегать, поскольку эту операцию пришлось бы повторять каждый раз после генерации новой версии. Оформление алфавитного указателя может варьироваться в широких пределах за счёт описанной в разделе 14.6 технологии настройки программы *MakeIndex*.

14.2. Процедура `theindex`

Процедура `theindex` печатает текст алфавитного указателя в две колонки[5]. Каждая запись в указателе начинается командой `\item`. В отличие от процедур составления списков, которые могут состоять только из записей одного уровня,

[3] Возможно, требуется обновление Windows Script Host до версии 5.6 или выше.
[4] Порядок сортировки отдельных терминов можно варьировать при помощи префиксной части *sortindex* аргумента команды `\index` (см. раздел 14.3). Однако подобное решение вряд ли удовлетворит авторов книг, подобной нашей, которая содержит более 7 000 входов в указатель.
[5] Ещё раз напомним, что подобные утверждения относятся к стандартным классам документов. В нашей книге алфавитный указатель напечатан в 3 колонки.

в процедуре `theindex` максимальное число уровней равно трём. Запись второго уровня начинается с команды `\subitem`, а запись третьего уровня — с `\subsubitem`. Пустые строки между записями игнорируются. Дополнительный вертикальный пробел, вставляемый перед записью, начинающей новую букву, создают при помощи команды `\indexspace`. *MakeIndex* автоматически вставляет её в нужных местах.

```
\begin{theindex}
  \item гну  24, 26, 97
  \item гнус 13, 43
  \subitem злой 14, 117
  \subsubitem очень злой 77
  \indexspace
  \item дикобраз 13
\end{theindex}
```

гну 24, 26, 97
гнус 13, 43
злой 14, 117
очень злой 77

дикобраз 13

Автору документа не придётся заполнять тело процедуры `theindex` «вручную», но он должен в исходном тексте пометить термины, из которых компилятору надлежит создать входы в указатель.

14.3. Вход в указатель

Алфавитный указатель должен помочь читателю найти то, что он ищет. Здравый смысл подскажет автору, что должно быть в указателе и как его следует организовать. Компьютер — всего лишь инструмент: он не может составить указатель за автора книги. Вероятно, не очень сложно выбрать, какие слова являются наиболее значимыми, и механически пометить каждое появление этих слов в тексте, но такой указатель вряд ли будет столь же полезен, как приготовленный с бо́льшим тщанием.

Каждая команда `\index` создаёт *вход* в алфавитный указатель. LaTeX записывает информацию о входе в файл *jobname*`.idx`, если преамбула входного файла содержит декларацию `\makeindex`. Например, если команда `\index{кит}` содержится в тексте, который в печатном документе попадает на страницу 42, то информация об этом в файле *jobname*`.idx` будет выглядеть следующим образом:

```
\indexentry{кит}{42}
```

Команда `\indexentry` служит входом в алфавитный указатель. Если в преамбуле нет декларации `\makeindex` (или же, напротив, имеется декларация `\nofiles`, запрещающая запись любых служебных файлов), то `\index` просто игнорируется. Команда `\index` не производит никакого эффекта в том месте входного файла, где она стоит. Во входном файле нужно набрать

```
Синий кит\index{кит}~--- самое крупное млекопитающее.
```

чтобы создать вход в алфавитный указатель к слову «кит». Дабы исключить ситуацию, когда команда `\index{кит}` попадает на страницу, следующую за той, где стоит слово «кит», она должна следовать за ним без пробела.

Следующий пример показывает, каким образом *MakeIndex* сортирует входы в указатель. В левой колонке указан номер страницы, куда попадает текст, содержащий команду `\index`. В правой колонке показан соответствующий фрагмент алфавитного указателя:

стр. ii	`\index{Альфа}`	алфа вит, 24
стр. viii:	`\index{альфа}`	Алфавит, ix
стр. ix:	`\index{альфа}`	алфавит, 23
	`\index{Алфавит}`	Альфа, ii
стр. 22:	`\index{альфа}`	альфа, viii, ix, 22
стр. 23:	`\index{алфавит}`	
	`\index{алфавит}`	
стр. 24:	`\index{алфа вит}`	

Здесь две команды `\index{алфавит}` на странице 23 генерируют номер 23 в алфавитном указателе только один раз.

Чтобы создать вход второго уровня в алфавитный указатель, аргумент команды `\index` должен содержать также указание на вход первого уровня. В этом случае два входа разделяются восклицательным знаком «`!`»:

стр. 7:	`\index{гну!полосатый}`	гну, 32
стр. 32:	`\index{гну}`	белохвостый, 35
стр. 35:	`\index{гну!белохвостый}`	полосатый, 7
	`\index{гнус!личинка}`	гнус
стр. 38:	`\index{гнус!куколка}`	куколка, 38
		личинка, 35

Вход третьего уровня должен указывать весь путь его поиска:

стр. 3:	`\index{укус!насекомого!комара}`	укус
стр. 4:	`\index{укус!насекомого!пчелы}`	насекомого
стр. 9:	`\index{укус!растения}`	комара, 3
		пчелы, 4
		растения, 9

MakeIndex и процедура `theindex` предусматривают не более трёх уровней в алфавитном указателе.

Чтобы выделить диапазон страниц, следует пометить начало и конец соответствующего фрагмента текста двумя командами `\index`. Аргумент первой команды должен заканчиваться парой символов `|(`, а второй — `|)`:

стр. vi:	`\index{гнус|(}`	гну
стр. x:	`\index{гнус|)}`	белохвостый, 28–32
стр. 22:	`\index{гнус}`	полосатый, 22
	`\index{гну!полосатый|(}`	гнус, vi–x, 22
	`\index{гну!полосатый|)}`	
стр. 28:	`\index{гну!белохвостый|(}`	
стр. 30:	`\index{гну!белохвостый}`	
стр. 32:	`\index{гну!белохвостый|)}`	

MakeIndex правильно обрабатывает случай, когда начало и конец диапазона попадают на одну страницу. Он также автоматически форматирует номера страниц в виде диапазона, если один и тот же вход в указатель встречается на трёх или более страницах подряд.

Иногда требуется добавить перекрёстную ссылку на другой термин в алфавитном указателе, не указывая номер страницы. Это достигается при помощи конструкции `see{...}`, которая отделяется от предшествующей части аргумента команды `\index` символом `|`:

стр. 2:	`\index{PC}`	PC, 2
стр. 2:	`\index{PC!IBM|see{IBM PC}}`	IBM, *см.* IBM PC

Если задать вход в указатель в форме *sortindex*`@`*printindex*, то строка *sortindex* определит расположение входа по алфавиту, а строка *printindex* — напечатанный текст. В следующем примере команда `$\alpha$` печатает символ α; соответствующий вход в указателе располагается после входа «альфа», а не перед входом β:

стр. 12:	`\index{альфы}`	β, 15
стр. 13:	`\index{альфа}`	альфа, 13
стр. 14:	`\index{альфа@$\alpha$}`	α, 14
стр. 15:	`\index{$\beta$}`	альфы, 12

Аналогично команда `\index{гнус@\textbf{гнус}}` создаёт вход «**гнус**». Более прямолинейное `\index{\textbf{гнус}}` приведёт к тому, что «**гнус**», как и «β», будет напечатан до слов на букву «а», поскольку программа сортировки считает, что обратный слеш, как и все иные подобные специальные символы, предшествует буквам.

В некоторых указателях определённые номера страниц форматируются особым способом. Например, номер страницы, где дано определение термина, может быть выделен курсивом. Курсивный текст печатает команда `\textit`. Её имя или имя другой команды *format*, которая должна печатать номер выделенной страницы, следует указать в самом конце аргумента команды `\index`, заменив обратный слеш \ на `|`. Номер страницы *pg*, где находится термин, помеченный командой `\index{...|`*format*`}`, будет напечатан в алфавитном указателе командой *format*{*pg*}. Аналогично команда `\index{...|(`*format*`}` поможет напечатать диапазон номеров страниц в форме *format*{*pg1*–*pg2*}. Например, что-

бы выделить ссылки на часть страниц курсивным и жирным шрифтом, следует действовать так:

стр. 3:	`\index{гну|textit}`	гну, *3*, **4**
стр. 4:	`\index{гну|textbf}`	гнус, 5, *44–46*
стр. 5:	`\index{гнус}`	
стр. 44:	`\index{гнус|(textit}`	
стр. 46:	`\index{гнус|)}`	

Конструкция `\see{...}`, упомянутая выше, является специальной реализацией этой возможности. Команда `\see` определена в пакете makeidx. Она имеет два аргумента, причём второй из них не печатает:

`\see{IBM, PC}{2}`	см. IBM, PC

Этим вторым аргументом является номер страницы, на которую попадает команда `\index`.

Другая подобная команда `\seealso` также «прячет» номер страницы, заменяя его словами «см. также». В отличие от `\see` она предназначена для выделения терминов, многократно помеченных в тексте документа. Соответственно, для таких терминов будут указаны номера страниц, сформированные другими командами `\index`:

стр. 5:	`\index{гнус}`	гнус, 5, *см. также* москиты
стр. 99:	`\index{гнус|seealso{москиты}}`	

Поскольку `\see` не печатает номер страницы, не имеет большого значения, где находятся команды `\index{...|see{...}}` (но они должны находиться в теле документа, т. е. между `\begin{document}` и `\end{document}`). Однако в случае `\seealso` положение `\index{...|seealso{...}}` определяет положение «см. также» относительно других номеров страниц. Обычно такого рода перекрёстные ссылки собирают в одном месте, чаще всего в конце документа.

Отладку алфавитного указателя в определённой степени способна облегчить загрузка пакета showidx. Он модифицирует команду `\index` так, что её аргумент печатается на полях страницы, тогда как другие функции команды сохраняются. Просматривая отпечатанную страницу с такими заметками на полях, легко установить, какие термины нужно добавить в алфавитный указатель или удалить из него.

14.3.1. Особые возможности

Команды в аргументе `\index` исполняются тогда, когда LaTeX форматирует алфавитный указатель, считывая файл с расширением `ind`, а не при записи информации в файл `job`*name*`.idx`[6].

[6] Хрупкие команды расшифровываются уже на стадии записи этого файла. Если это нежелательно, такие команды необходимо защитить, предварив их командой `\protect`.

Следовательно, *MakeIndex* расставляет по алфавиту входы в указатель, записанные командой \index{\xyz} в соответствии с написанием имени команды \xyz (учитывая даже «\») вне зависимости от того, как она определена.

Аргумент команды \index может содержать любые символы, в том числе специальные. Если аргумент команды \index или его левая часть, стоящая до символа @ (при его наличии), содержит любой из десяти специальных символов: \, #, $, %, &, {, }, _, ^ и ~, то она не может находиться в аргументе другой команды. Однако и в этом случае устойчивые команды могут быть помещены в части аргумента команды \index, следующей за символом @, как, например, в \index{гну@\textit{гну}}, а хрупкие команды могут использоваться, если защищены командой \protect.

Фигурные скобки в аргументе \index всегда должны быть парными, причём скобки в командах \{ или \} также учитываются. Впрочем, команды \{ и \} имеют синонимы \lbrace и \rbrace, на которые это ограничение уже не распространяется.

Символы !, @ и | имеют значение команд, если они находятся в аргументе команды \index. Для использования в качестве обычных символов их следует процитировать, поставив перед каждым из них двойную кавычку ". И вообще, любой символ считается цитированным, если он следует за нецитированным символом ", не являющимся частью команды \".

стр. 2:	\index{восклицать ("!)}	восклицать (!), 2
стр. 3:	\index{восклицать ("!)!громко}	громко, 3
стр. 4:	\index{еж@\"{е}ж}	ёж, 4
стр. 5:	\index{кавычка ("")}	кавычка ("), 5

В процессе упорядочения входов по алфавиту *MakeIndex* рассматривает пробелы наравне с обычными символами. Поэтому команды \index с аргументами {гну}, {␣гну} и {гну␣}, где ␣ обозначает пробел, создают три разных входа в указатель. Однако в печатном документе все три входа выглядят совершенно одинаково, так как LaTeX игнорирует пробелы при печати. Точно так же команды \index{a␣space} и \index{a␣␣space} создают два разных входа, которые неразличимы при печати. Поэтому нельзя переносить часть аргумента команды \index на другую строку во входном файле. Эти проблемы с пробелами можно устранить, если программу *MakeIndex* запускать с ключом -c.

MakeIndex предполагает, что страницы, нумерованные римскими цифрами, предшествуют страницам с арабской нумерацией, а страницы, помеченные латинскими буквами, находятся в самом конце книги. Можно изменить это правило, переопределив параметр page_precedence в стилевом файле, как показано в разделе 14.6.

14.4. Два указателя в одном документе

Изредка случается, что в книге одновременно нужны алфавитный указатель и словарь терминов или предметный и именной указатели. Разберем последний случай. Как мы отмечали, подготовку словаря терминов чаще всего можно выполнить без привлечения сортирующей программы.

Для выделения терминов, идущих в именной указатель, можно использовать команду \glossary, оставив \index для предметного указателя. При наличии в преамбуле декларации \makeglossary информация о таких терминах будет записана в служебный файл *jobname*.glo. Он состоит из *входов в словарь терминов*, т. е. из команд \glossaryentry (в аргументах которых записана информация

о терминах), тогда как файл *jobname*.idx для предметного указателя состоит из команд \indexentry. Термин «вход в словарь терминов» не должен вводит Читателя в заблуждение — это всего лишь буквальный перевод имени команды \glossaryentry. В данной книге мы используем \glossary для составления именного указателя.

MakeIndex по умолчанию настроен на сортировку файла с расширением idx. Для сортировки файла с расширением glo, содержащего \glossaryentry вместо \indexentry, нужно создать стилевой файл для программы *MakeIndex* (назовём его *manual-g*.ist) примерно следующего содержания:

```
keyword         "\\glossaryentry"
heading_prefix  "{\\bfseries\\hfill "
heading_suffix  "\\hfill\\nopagebreak\n}"
delim_0          "\\quad "
delim_1          "\\quad "
delim_2          "\\quad "
```

Смысл того, что здесь написано, поясняется в разделе 14.6. Чтобы первичный «словарь терминов» (то бишь алфавитный указатель) *jobname*.glo сохранить в файле *jobname*.gls, командная строка запуска программы *MakeIndex* должна содержать ключи[7] -s и -o:

```
cscript cyr2win.wsf jobname.glo
makeindx.exe -s manual-g.ist -o jobname.gls jobname.glo
```

Затем файл *jobname*.gls нужно вставить в печатный документ при помощи команды

```
\input{jobname.gls}
```

В результате указанных действий «словарь терминов» будет напечатан процедурой theindex. Заметим, что команда \printindex, рекомендуемая для печати алфавитного указателя, фактически эквивалентна \input{*jobname*.ind}, но не выдаёт сообщение об ошибке, если файл *jobname*.ind не найден.

14.5. Подведём итоги

Компиляцию входов в алфавитный указатель осуществляют следующие команды:

\makeindex — декларация, располагающаяся только в преамбуле. При наличии \makeindex и отсутствии декларации \nofiles LaTeX записывает в файл *jobname*.idx входы в указатель, т. е. \indexentry, произведённые командами \index.

[7] Полное описание всех ключей программы *MakeIndex* содержится в прилагаемой к ней документации [19] и в книге [12].

⚠ `\index{str}` записывает в файл *jobname*.`idx` (если запись разрешена) вход в указатель `\indexentry{`*str*`}{`*pg*`}`, где *pg* — номер страницы. Строка входа *str* может содержать любые символы, включая специальные, но не должна иметь непарные фигурные скобки, при подсчёте которых учитываются команды `\{` и `\}`. Структура строки входа *str* в наиболее общем случае имеет следующий вид:

$sortindex_0$`!`$sortindex_1$`!`$sortindex_2$`@`*printindex*`|`*format*

где только первый элемент строки $sortindex_0$ является обязательным. Элементы $sortindex_0$, `!`$sortindex_1$ и `!`$sortindex_2$ используются для сортировки в алфавитном порядке соответственно входов первого уровня, входов второго уровня внутри входа первого уровня и входов третьего уровня внутри входа второго уровня, причём `!` является разделителем между этими элементами и не учитывается при сортировке. Элемент *printindex* есть текст входа наинизшего уровня, имеющегося в *str*. При отсутствии `@`*printindex* текст входа соответствующего уровня совпадает с $sortindex_0$, $sortindex_1$ или $sortindex_2$. Последняя часть строки `|`*format* определяет способ форматирования номера страницы *pg* в алфавитном указателе, причём форматирование осуществляет команда `\`*format*`{`*pg*`}`, которая должна быть определена к моменту печати указателя. Команда `\`*format*`{`*pg*`}` может иметь более одного аргумента; здесь предполагается, что дополнительные аргументы включены в обозначение *format*. Вместо `|` могут стоять `|(` или `|)`. Команда `\index` с `|(` помечает начало фрагмента текста, который должен быть отражён в алфавитном указателе в виде диапазона страниц, а команда `\index` с `|)` помечает конец этого фрагмента текста, причём части аргументов команд `\index`, предшествующие `|`, должны совпадать.

Команда `\index`, помимо того что является хрупкой, не может появляться внутри аргумента другой команды, если $sortindex_0$, $sortindex_1$ или $sortindex_2$ содержат символы, отличные от букв, цифр и знаков препинания.

`\see{`*text*`}{`*pg*`}` определена в пакете makeidx. Используется для создания перекрёстных ссылок на другие входы в алфавитный указатель. Обращение к `\see` из аргумента команды `\index` должно иметь вид `\index{...|see{`*printindex*`}}`.

`\seealso{`*text*`}{`*pg*`}` определена в пакете makeidx и действует аналогично команде `\see`. Обращение к `\seealso` из аргумента команды `\index` должно иметь вид `\index{...|seealso{`*printindex*`}}`.

Компиляцию входов в словарь терминов осуществляют следующие команды:

`\makeglossary` — декларация, располагающаяся только в преамбуле. Если отсутствует `\nofiles`, `\makeglossary` заставляет LaTeX записывать в файл *jobname*.`glo` входы в словарь терминов `\glossaryentry`, произведённые командами `\glossary`.

⚠ `\glossary{str}` записывает в файл *jobname*`.glo` (если запись разрешена) вход в словарь терминов `\glossaryentry{str}{pg}`, где *str* и *pg* имеют тот же смысл, что и для команды `\index`. Команда `\glossary` не может появляться внутри аргумента другой команды, если $sortindex_0$, $sortindex_1$ или $sortindex_2$ содержат символы, отличные от букв, цифр и знаков препинания.

Команда

`\printindex` (makeidx)

определена в пакете makeidx. Она считывает файл *jobname*`.ind`, куда *MakeIndex* записывает отсортированный текст алфавитного указателя, оформленный в виде процедуры `theindex`.

Процедура

⚠ `\begin{theindex} ... \end{theindex}`

печатает алфавитный указатель. Каждый вход первого, второго и третьего уровня начинается с команды

`\item   \subitem   \subsubitem`

соответственно. Они, в отличие от команды `\item` в других процедурах составления списков (раздел 5.4), не имеют аргументов. Команда

`\indexspace`

вставляет дополнительный вертикальный пробел между входами в алфавитный указатель, обычно используется перед входом, начинающимся с новой буквы. Величина пробела, как и другие детали оформления алфавитного указателя, определяется классом печатного документа. Название алфавитного указателя и ключевые слова, вставляемые командами `\see` и `\seealso`, хранятся соответственно в командах

`\indexname`
`\seename   \alsoname` (babel, makeidx)

Они переопределяются при помощи `\renewcommand`. Например:

```
\renewcommand{\indexname}{Предметный указатель}
\renewcommand{\alsoname}{\emph{см.~тж.}}
```

14.6. Настройка указателя

Формат алфавитного указателя не фиксирован и может быть адаптирован к конкретной задаче. Например, легко заменить символ `!`, который используется для разделения записей разных уровней в алфавитном указателе. То же самое можно проделать с другими служебными символами `@`, `|`, `"`, `{`, `}` и т. п. Иными словами, кардинальную переделку допускает формат входного

файла с расширением idx для программы *MakeIndex*. *MakeIndex* способен также в широких пределах варьировать формат выходного файла с расширением ind, который содержит исходный текст алфавитного указателя для окончательной компиляции LaTeX'ом. Тем самым исключается потребность в ручной доводке указателя. Настройкой *MakeIndex* управляет стилевой файл (с расширением ist). Он состоит из набора параметров.

Все параметры подразделяются на две группы. В первую входят параметры, управляющие форматом входного файла; они перечислены в табл. 14.1. Вторую группу составляют настройки выходного файла; они представлены в табл. 14.2. В каждой таблице приведено значение параметра настройки, принимаемое по умолчанию (когда параметр отсутствует в стилевом файле или стилевой файл вообще не используется). Именно эти значения подразумевались в предшествующих разделах главы. Пары параметр — значение могут располагаться в стилевом файле в произвольном порядке. Строка, начинающаяся со знака процента %, является комментарием. Изменяемое значение параметра, обозначенное в таблицах через (s), есть произвольная строка символов, ограниченная с обеих сторон двойными апострофами ("..."); (c) обозначает отдельную букву в одинарных апострофах, а (n) — целое число, \n и \t — переход на новую строку и табулятор в выходном файле[8]. Чтобы ввести в строку символов обратный слеш или кавычки, необходимо перед ними поставить escape-символ (обратный слеш).

В качестве первого упражнения рассмотрим стилевой файл для подготовки словаря терминов. Назовем его myglossary.ist и предположим, что термины для словаря помечены в исходном тексте командами \glossary:

```
keyword   "\\glossaryentry"
preamble  "\\newpage
          {\\Large \\bfseries Словарь терминов}
          \\begin{description}\n"
postamble "\n\n\\end{description}\n"
```

В первой строке он содержит указание, что входы в словарь терминов записываются в аргумент команды \glossaryentry, а не \indexentry, как программа *MakeIndex* предполагает по умолчанию. Далее в трёх строчках идет описание преамбулы выходного файла. Здесь указывается, что словарь терминов начинается с новой страницы (\newpage). Затем формируется заголовок словаря терминов, который должен быть напечатан большими буквами полужирным шрифтом (\Large и \bfseries). Для форматирования словаря терминов выбрана процедура description. У неё могут быть записи только одного уровня. Поэтому получается, что в аргументах команд \glossary, которые метят термины в исходном тексте печатного документа, нельзя использовать символ !, если ему не предшествует двойной апостроф ". Если исходный текст документа записан в файле с именем foo.tex, то командная строка запуска программы *MakeIndex* должна выглядеть так:

```
makeindex  -s myglossary.ist  -o foo.gls  foo.glo
```

Следующий пример стилевого файла (назовем его book.ist) показывает, как напечатать алфавитный указатель в виде отдельного документа.

```
preamble
 "\\documentclass[12pt]{book}
 \\begin{document}
 \\begin{theindex}
 {\\small\n"
```

[8] Форматирование текста в выходном файле программы *MakeIndex* не влияет на результат компиляции алфавитного указателя LaTeX'ом.

```
postamble
 "\n\n}
 \\end{theindex}
 \\end{document}\n"
```

Если предположить, что алфавитный указатель должен начинаться на нечётной странице, запуск *MakeIndex* осуществляется из командной строки с дополнительным ключом `-p odd`:

```
makeindex  -s book.ist  -o footmp.tex  -p odd  foo
```

MakeIndex извлечёт номер последней страницы исходного документа из файла протокола компиляции (в данном случае это `foo.log`) и возьмёт следующее ближайшее нечётное число. Другие возможные значения ключа `-p` устанавливают, что первая страница должна быть нечётной (`even`) или просто иметь номер на 1 больше, чем в исходном документе (`any`). Можно также явно указать номер страницы в виде числа. Ключ `-o` направляет вывод программы *MakeIndex* в файл `footmp.tex`, полностью готовый к обработке LaTeX'ом.

Наконец, приведём текст стилевого файла `manual-i.ist`, который использовался для подготовки алфавитного указателя в нашей книге:

```
headings_flag   1
symhead_positive "Символы"
numhead_positive "Числа"
heading_prefix "{\\bfseries\\uppercase{"
heading_suffix "} \\hrulefill\\nopagebreak\n}"
```

В первой строке сказано, что нужно создавать заголовки групп терминов и начинать их с прописной буквы. Вторая и третья строки заменяют английские названия для групп Symbols (символы) и Numbers (числа). Следующие две строчки устанавливают, что заголовок группы надо печатать полужирным шрифтом и затем проводить горизонтальную линию до конца колонки[9].

Все возможные параметры, через которые можно воздействовать на оформление отсортированного указателя, перечислены в табл. 14.1.

Таблица 14.1

Параметры входного файла *MakeIndex*

Параметр	Значение по умолчанию и описание
1	2
`keyword` (s)	`"\\indexentry"` Команда, аргумент которой является входом в алфавитный указатель
`arg_open` (c)	`'{'` Признак начала аргумента команды-входа в указатель
`arg_close` (c)	`'}'` Признак конца аргумента команды-входа в указатель
`range_open` (c)	`'('` Признак начала диапазона страниц

[9] Дополнительно мы изменили определения команд `\subitem` и `\subsubitem` так, чтобы они вставляли соответственно одно и два тире перед записями второго и третьего уровня, а также процедуру `theindex` так, чтобы она печатала указатель в три колонки. Эти изменения сделаны в классе печатного документа. Их описание выходит за рамки книги, так как требует знания команд TeX'а. Читатель найдёт необходимые сведения в книге [12].

Продолжение табл. 14.1

1	2
`range_close` (c)	`')'` Признак конца диапазона страниц
`level` (c)	`'!'` Признак начала следующего уровня во входе в указатель
`actual` (c)	`'@'` Признак начала текста, который должен появиться в выходном файле
`encap` (c)	`'|'` Признак начала текста, который используется как имя команды, форматирующей номер страницы
`quote` (c)	`'"'` Признак цитируемого символа. Следующий символ теряет свойство быть признаком чего-либо и рассматривается в качестве обычной буквы
`escape` (c)	`'\\'` Символ, являющийся признаком команды, если ему не предшествует другой символ `escape`; `quote` теряет свойство быть признаком цитируемого символа, если ему предшествует `escape`. Эти два символа должны быть разными

Таблица 14.2

Параметры выходного файла *MakeIndex*

Параметр	Значение по умолчанию и описание
1	2
Контекст	
`preamble` (s)	`"\\begin{theindex}\n"` Текст, предшествующий отсортированному списку входов в алфавитный указатель
`postamble` (s)	`"\n\n \\end{theindex}\n"` Текст, следующий за отсортированным списком входов в алфавитный указатель
Начальная страница	
`setpage_prefix` (s)	`"\n \\setcounter{page}{"` Начало команды, которая задаёт номер первой страницы (используется, если алфавитный указатель компилируется отдельно). Номер первой страницы *pg* может быть задан ключом `-p` *pg* в командной строке запуска программы *MakeIndex*
`setpage_suffix` (s)	`"}\n"` Конец команды, которая задаёт номер первой страницы (используется, если алфавитный указатель компилируется отдельно)

Продолжение табл. 14.2

1	2
Новая группа или буква	
`group_skip` (s)	`"\n\n \\indexspace\n"` Вертикальный пробел, который нужно вставить перед группой терминов, начинающихся с нового символа
`headings_flag` (n)	`0` Флаг, указывающий способ формирования заголовка новой группы терминов. Если $n > 0$, в заголовок помещается прописная буква; если $n < 0$ — строчная буква; при $n = 0$ заголовок не создаётся
`heading_prefix` (s)	`""` Текст, который вставляется перед заголовком группы
`heading_suffix` (s)	`""` Текст, который вставляется после заголовка группы
`symhead_positive` (s)	`"Symbols"` Заголовок группы символов в случае, если `headings_flag` > 0
`symhead_negative` (s)	`"symbols"` Заголовок группы символов в случае, если `headings_flag` < 0
`numhead_positive` (s)	`"Numbers"` Заголовок группы цифр, если `headings_flag` > 0
`numhead_negative` (s)	`"numbers"` Заголовок группы цифр, если `headings_flag` < 0
Разделители уровней	
`item_0` (s)	`"\n \\item "` Команда, которая вставляется перед входом первого уровня
`item_1` (s)	`"\n \\subitem "` Команда, которая вставляется между двумя входами второго уровня
`item_2` (s)	`"\n \\subsubitem "` Команда, которая вставляется между двумя входами третьего уровня
`item_01` (s)	`"\n \\subitem "` Команда, которая вставляется между входом первого и второго уровня
`item_x1` (s)	`"\n \\subitem "` Команда, которая вставляется между входом первого и второго уровня, когда первый не имеет номера страницы
`item_12` (s)	`"\n \\subsubitem "` Команда, которая вставляется между входом второго и третьего уровня
`item_x2` (s)	`"\n \\subsubitem "` Команда, которая вставляется между входом второго и третьего уровня, когда второй уровень не имеет номера страницы

Продолжение табл. 14.2

1	2
Разделители страниц	
`delim_0` (s)	`", "` Разделитель, который вставляется между термином и первым номером страницы
`delim_1` (s)	`", "` То же для второго уровня
`delim_2` (s)	`", "` То же для третьего уровня
`delim_n` (s)	`", "` Разделитель, который вставляется между номерами страниц для одного термина на любом уровне
`delim_r` (s)	`"--"` Разделитель, который вставляется между номерами первой и последней страниц диапазона
`delim_t` (s)	`""` Разделитель, который вставляется вслед за последним номером в списке страниц
Формат и порядок номеров страниц	
`encap_prefix` (s)	`"\\"` Первая часть префикса для команды, форматирующей номер страницы
`encap_infix` (s)	`"{"` Вторая часть префикса для команды, форматирующей номер страницы
`encap_suffix` (s)	`"}"` Суффикс для команды, форматирующей номер страницы
`page_precedence` (s)	`"rRnaA"` Порядок сортировки страниц с разным форматом номеров: `r`, `R` — римские строчные и прописные цифры; `n` — арабские цифры; `a`, `A` — строчные и прописные буквы
Перенос строк	
`line_max` (n)	`72` Максимальная длина строки в выходном файле программы *MakeIndex*. Более длинный текст разбивается на несколько строк
`indent_space` (s)	`"\t\t"` Отступ, который вставляется перед перенесенной строкой в выходном файле программы *MakeIndex* (по умолчанию вставляются два табулятора)
`indent_length` (n)	`16` Длина отступа, который вставляется перед перенесённой строкой в выходном файле программы *MakeIndex* (по умолчанию вставляются 16 пробелов, что эквивалентно двум табуляторам)

Выпустив джина из бутылки,
возьмите ёмкость побольше,
чтобы упаковать его вновь.
Первый закон эволюции

Глава 15

Классы документов

До сих пор мы мало обращали внимание на различия, существующие между классами печатных документов. Бо́льшая часть из того, что Читатель уже знает о LaTeX'е, в равной степени применима к печатному документу любого из шести стандартных классов `article`, `proc`, `book`, `report`, `slides`, `letter`. Время от времени мы указывали на какие-то малозначительные различия, но теперь поговорим о классах печатных документов более обстоятельно. О классе `article`, предназначенном для подготовки статей, пожалуй, уже всё сказано. Возьмём его за основу и посмотрим, чем примечательны другие классы. В завершение главы в качестве примера нестандартного класса мы рассмотрим класс `revtex4`, который используют многие зарубежные и отечественные научные журналы.

Напомним, что выбор класса печатного документа выполняет декларация `\documentclass`, которая открывает преамбулу. Например, для подготовки слайдов входной файл следует начать с

```
\documentclass[a4paper,landscape]{slides}
```

причём выбор опций `a4paper` и `landscape` предполагает, что слайды будут напечатаны на листах формата A4 в альбомной ориентации.

15.1. Класс `proc`

Класс `proc` предназначен для подготовки тезисов докладов на научных конференциях. Он создан на основе класса `article`, отличаясь тем, что текст печатается в две колонки. Некоторые опции (и в их числе `onecolumn`) не поддерживаются, так что напечатать текст в одну колонку не получится. Вот и всё!

15.2. Класс `book`

Книги (класс `book`) отличаются от отчётов (класс `report`), главным образом, оформлением вводной и заключительной частей. Вводная часть книги обычно включает авантитул, контртитул, титульную страницу (титул), оборот титула,

предисловие и оглавление. Она может также содержать посвящение, список рисунков, список книг той же серии и т. п. Заключительная часть может содержать послесловие, алфавитный указатель, список литературы и т. д.

Класс book не пытается сопоставить специальную команду или процедуру каждой структуре, которая может появиться во вводной или заключительной части книги. В этом смысле он не отличается от класса report. Отличие создают декларации

```
\frontmatter
\mainmatter
\backmatter
```
(book)

Предполагается, что первая из них должна открывать вводную часть книги, вторая — основную, третья — заключительную. Во вводной части книги страницы иногда нумеруются римскими цифрами, а в основной и заключительной частях — арабскими. Именно такого рода перенастройку выполняют \frontmatter, \mainmatter и \backmatter. Во вводной и заключительной частях книги команда \chapter не печатает номер главы, но вставляет её заголовок в оглавление. Поэтому предисловие можно начать строкой

```
\frontmatter \chapter{Вместо предисловия}
```

Другие команды секционирования действуют одинаковым образом в любой части книги; только их *-форма подавляет печать номера раздела (но она не вставляет заголовок в оглавление).

Двусторонняя печать является нормой для книг, поэтому по умолчанию класс book использует опцию twoside вместо oneside. Также по умолчанию используется опция openright, и в результате каждая глава начинается с правой (нечётной) страницы. Напротив, для класса report (как и article) по умолчанию действует опция openany.

Книга может иметь более одной титульной страницы, а каждая титульная страница может содержать много разной информации помимо названия и фамилий авторов — например, название издательства или серии, фамилию оформителя и т. п. Команда \maketitle поэтому малополезна при оформлении книги. Чтобы достичь желаемого вида её титульной страницы, возможно, придётся потратить немало усилий и времени. В связи с этим полезно будет ознакомиться с главой 17 или освежить в памяти содержание глав 9–10.

15.3. Класс slides

Понятие *слайд* традиционно связывают с позитивным изображением на фотоплёнке. Плёнку разрезали на кадры и вставляли в рамочки из бумаги или пластмассы, а полученные слайды проецировали на экран с помощью диапроектора. Затем были придуманы специальные аппараты *оверхэды* (от английского слова *overhead*), проецирующие на экран текст, написанный на прозрачной плёнке

размером с обычный лист писчей бумаги. Такие прозрачные листы иногда называют *прозрачками*, но мы будем применять проверенный временем термин *слайды*. После появления специальной прозрачной плёнки для принтеров изготовление слайдов значительно упростилось, а их использование стало обычным делом для докладчиков на научных конференциях. С появлением аппаратов, способных проецировать изображение с экрана монитора на большой экран, слайды стали делать в «электронном виде», отказавшись от вывода на пленку. Теперь профессора даже лекции для студентов читают, показывая заготовленные иллюстрации с экрана компьютера.

Чтобы изготовить слайды с помощью LaTeX'а, следует выбрать класс `slides`. И если слайды предназначены для показа на конференции или на лекции, то лучше всего компилировать исходный текст сразу в формат PDF с помощью программы `pdflatex`, поскольку средства просмотра документов PDF имеет практически любой компьютер.

Создание хороших слайдов требует скорее визуального, а не логического подхода к их проектированию. В этом смысле LaTeX не очень пригоден для такой работы. Однако имеется несколько доводов в пользу LaTeX'а и в этом случае. LaTeX следует предпочесть, если

- слайды основываются на материале из документа, подготовленного с помощью LaTeX'а;
- текст содержит много математических формул;
- слайды приходится делать не столь часто, чтобы изучать иную издательскую систему.

Для слайдов используют шрифты большого размера: шрифт `\small` в классе `slides` выглядит, как `\Large` в обычном печатном документе. Это сделано специально, чтобы слайды было легко читать издали.

Класс `slides` имеет почти стандартный набор опций (раздел 3.2). Не реализованы опции `twocolumn` (печать в две колонки) и `twoside` (двусторонняя печать). Добавлена опция `clock`, о которой мы расскажем ниже.

15.3.1. Слайды и оверлеи

Отдельные слайды производятся процедурой `slide`:

`\begin{slide} ... \end{slide}` (slides)

Она формирует отдельную страницу, как показывает следующий пример[1].

[1] Примеры в данном разделе получены уменьшением страниц формата А4 при одновременном увеличении кегля в 1,6 раза.

```
\begin{slide}
 \begin{center} МЫШИ и ЛИСЫ \end{center}
    это животные, живущие в норах.

    Мыши маленькие.

    Лисы большие.
\end{slide}
```

МЫШИ и ЛИСЫ

это животные, живущие в норах.

Мыши маленькие.

Лисы большие.

1

Здесь и ниже в данном разделе предполагается, что класс `slides` загружен с опцией `landscape`, как показано на стр. 341. Текст на слайде производится обычными командами LaTeX'а. Можно использовать любые команды, которые имеют смысл при изготовлении слайдов. Лишены смысла команды секционирования, поскольку каждый слайд есть логически завершенный раздел доклада. Не определены процедуры `figure` и `table`, поскольку изготовитель слайдов берет на себя заботу о размещении больших иллюстраций. Так же не имеют смысла команды прерывания страницы `\newpage`, `\clearpage` и т. д., так как каждый слайд печатается на отдельном листе. Чтобы сделать цветные слайды, нужно использовать пакет color (раздел 10.7). Он загружается в преамбуле декларацией `\usepackage`.

Оверлеями называются составные слайды, которые накладывают друг на друга, формируя изображение из отдельных частей. Оверлеи делает процедура

`\begin{overlay} ... \end{overlay}` (slides)

Она аналогична `slide`, только нумерует слайды иным способом. Например, оверлеи, следующие за слайдом номер 4, будут иметь номера 4-а, 4-б и т. д. Наилучший способ изготовить оверлейные слайды, которые точно совмещаются при наложении, состоит в том, чтобы использовать один и тот же текст для всех оверлеев, но сделать отдельные его части невидимыми. Для этого удобно использовать пакет color, а невидимый текст напечатать белым цветом.

```
\begin{slide}
\begin{center}
    МЫШИ \textcolor{white}{и ЛИСЫ}
\end{center}
    это животные, живущие в норах.

    Мыши маленькие.

    \textcolor{white}{Лисы большие.}
\end{slide}
```

МЫШИ

это животные, живущие в норах.

Мыши маленькие.

2

```
\begin{overlay}
 \begin{center}
    \textcolor{white}{МЫШИ} и ЛИСЫ
 \end{center} \textcolor{white}
 {это животные, живущие в норах.}

 \textcolor{white}{Мыши маленькие.}

Лисы большие.
\end{overlay}
```


Вместо пакета color можно использовать команды `\hspace` и `\vspace`, чтобы зарезервировать пустое место на слайде.

15.3.2. Заметки на слайдах

Для лекций или докладов удобно делать вспомогательные слайды с напоминанием о том, что нужно сказать. Процедура

`\begin{note} ... \end{note}` (slides)

производит заметку на одну страницу. Некоторые докладчики для каждого слайда делают одну-две страницы пояснений. Заметки, следующие за слайдом 5, будут иметь номера 5-1, 5-2 и т. д., причём можно печатать только те слайды или заметки, которые действительно нужны (см. следующий раздел).

Время доклада обычно ограничено. Поэтому на слайдах можно указывать необходимый период демонстрации данного слайда. Для этих целей в классе `slides` существует опция `clock`. При её наличии в `\documentclass` команда

`\addtime{`*secs*`}` (slides)

позволяет следить за графиком выступления. Сразу после каждого слайда нужно вставить команду `\addtime`, указывающую, сколько секунд *secs* планируется потратить на этот слайд. Внизу заметки, следующей за слайдом, будет напечатано полное время в минутах, которое по расчётам должно быть потрачено к этому моменту.

```
\documentclass[a4paper,landscape,clock]
    {slides}
\begin{document}
\begin{slide} ... \end{slide} \addtime{120}
\begin{slide} ... \end{slide} \addtime{180}
\begin{slide} ... \end{slide} \addtime{60}
\begin{note}
 Уничтожение комаров распылением дуста
 в Новосибирском Академгородке привело
 к исчезновению птиц в окрестных лесах.
\end{note}
```

После компиляции всего документа LaTeX выведет на экран монитора суммарное время, которое планируется потратить на демонстрацию слайдов. Полное время можно переустановить командой

`\settime{`*secs*`}` (slides)

Например, `\settime{180}` приравнивает его к 180 секундам. Не следует использовать команды `\addtime` или `\settime` внутри процедур `slide`, `note` или `overlay`.

15.3.3. Печать выбранных слайдов и заметок

`\onlyslides{`*slide-list*`}`
`\onlynotes{`*note-list*`}` (slides)

Если какие-то слайды пришлось исправить, то необязательно перепечатывать все. Можно откомпилировать (и затем напечатать) только те слайды, которые необходимы. Декларация

```
\onlyslides{1,6-9,31}
```

в преамбуле входного файла заставит LaTeX сгенерировать слайды 1, 6, 7, 8, 9, 31 и все их оверлеи. Номера слайдов в аргументе *slide-list* должны стоять в возрастающем порядке, причём допускаются номера несуществующих слайдов. Например,

```
\onlyslides{9-999}
```

не напечатает первые 8 слайдов. Аргумент `{`*slide-list*`}` декларации `\onlyslides` не должен быть пуст.

Аналогичным образом работает декларация `\onlynotes`. После

```
\onlynotes{9}
```

будут напечатаны все заметки 9-1, 9-2 и т. д. к слайду номер 9. Если во входном файле имеется только `\onlyslides`, но отсутствует `\onlynotes`, то будут воспроизведены только слайды без заметок. Использование обеих деклараций позволит напечатать и слайды, и заметки.

15.3.4. Текст вне процедур

Входной файл может содержать текст вне процедур `slide`, `overlay` или `note`. Такой текст будет воспроизводиться так же, как на слайдах, но без нумерации страниц. Таким образом можно сформировать титульную страницу. Текст вне процедур будет напечатан даже при использовании деклараций `\onlyslides` и `\onlynotes`. Чтобы разбить такой текст на слайды нужного объёма, можно использовать команду `\pagebreak`, `\newpage` и т. д.

15.4. Класс `letter`

Класс `letter`, используемый для написания писем, имеет ряд существенных отличий от других стандартных классов. С одной стороны, этот класс определяет ряд команд, не доступных другим классам, поскольку эпистолярный жанр являет собой особый вид искусства. С другой стороны, в нём отсутствуют процедуры `figure` и `table`, команды секционирования и блокирована возможность форматирования текста в две колонки стандартными средствами LaTeX'а (опция `twocolumn` не работает). У писем не бывает титульной страницы, поэтому не определены команда `\maketitle`, процедура `titlepage`, а у класса `letter` нет опции `titlepage`.

Письма могут быть деловыми (официальными) или частными (неофициальными). Текст письма должен предваряться перечисленными ниже декларациями. Они объявляют имя, адрес и другие атрибуты отправителя, которые, вероятно, одинаковы для всех писем. В одном входном файле можно записать несколько писем, а атрибуты отправителя достаточно указать только один раз. В таком случае эти декларации обычно помещают в преамбулу. Однако они могут появляться в любом месте входного файла, так как область их действия подчиняется обычным правилам (раздел 2.5).

`\address{...}` задаёт обратный адрес отправителя письма для конверта и заголовка письма. Отдельные строки адреса должны разделяться командами `\\`. Например:

```
\address{630090 Новосибирск \\ пр. Науки 99\\ Институт Колдовства}
```

Если декларацию `\address` пропустить, то письмо будет оформлено как официальное, которое обычно печатают на фирменном бланке, содержащем все атрибуты отправителя. LaTeX оставит на первой странице письма пустое место там, где обычно на фирменном бланке напечатан обратный адрес. Если же `\address` имеется, то письмо будет оформлено как частное.

`\signature{...}` задаёт текст подписи отправителя. Этот текст будет напечатан после команды `\closing` (см. ниже). Он может включать указание на должность и фамилию отправителя. Например:

```
\signature{профессор Р.\,А.~Зумов}
```

Менее формальную часть подписи печатает команда `\closing` (см. ниже).

`\name{...}` определяет атрибуты отправителя в обратном адресе, который печатается на конверте (см. `\makelabels` ниже). Хотя отправляет и подписывает письмо обычно (но не всегда) один и тот же человек, в почтовом адресе на конверте отправитель может быть указан иначе, чем в тексте письма:

```
\name{Проф. Р.\,А.~Зумов}
```

Если декларация `\signature` пропущена, письмо будет подписано так, как указано в `\name`.

`\location{...}` уточняет стандартный адрес отправителя, если письмо печатается на официальном бланке. Например, к адресу фирмы может быть добавлен номер кабинета конкретного отправителя письма. Если письмо частное (имеется `\address`), то уточнение адреса бессмысленно, поэтому данная декларация попросту игнорируется.

`\telephone{...}` определяет номер телефона отправителя, который указывается на бланке, если письмо является официальным. В частном письме номер телефона отправителя целесообразно указывать в адресе, поэтому данная декларация игнорируется.

Перечисленные декларации подготавливают информацию для процедуры `letter`, которая собственно и форматирует текст письма, поэтому они должны располагаться перед командой `\begin{letter}`. Процедура

`\begin{letter}{`*adr*`}` *text* `\end{letter}` (letter)

имеет аргумент *adr*, где можно указать адрес получателя. Текст письма открывается командой

`\opening{...}` (letter)

аргумент которой должен содержать обращение к адресату, а заканчивается командой

`\closing{...}` (letter)

Например,

```
\begin{letter}{Начальнику Департамента Хиромантии\\
    Его сиятельству Кн. В.\,А.~Длиннорукому\\
    в собственные руки\\ С.-Петербург}
\opening{Милостивый государь\\князь...}
    ...
\closing{Вашего высокопревосходительства\\ покорнейший слуга...}
\end{letter}
```

Как обычно, любой аргумент может быть пустым. Например, если пуст аргумент команды `\opening`, то обращение к милостивому государю перед текстом письма не появится. Тогда это обращение нужно включить в текст письма. В теле процедуры `letter` перед командой `\opening` можно вставить любые декларации, в том числе изменяющие адрес, имя, подпись отправителя и т. д. В этом случае они будут действительны только для данного письма.

Команды секционирования (раздел 3.5) и команды форматирования титульной страницы (раздел 3.4) не определены в классе `letter`.

После команды `\closing` можно добавить ещё небольшую порцию текста, однако команда `\closing` запрещает перенос текста на следующую страницу и устанавливает отступ в начале каждого абзаца равным нулю. Декларация

`\ps` (letter)

восстанавливает нормальное форматирование текста, следующего за командой `\closing`. Её используют для печати постскриптума, однако знак постскриптума «P.S.» не печатается; если нужно, его следует вставить в текст. Команда

`\cc{cop}` (letter)

печатает список адресатов, которым направлены копии письма, а команда

`\encl{list}` (letter)

печатает список вложений. Элементы списков cop и list отделяются командами `\\`.

Адрес получателя письма можно дополнительно напечатать на отдельном листе (или прямо на конверте), если в преамбулу (перед `\begin{document}`) поместить декларацию

`\makelabels` (letter)

В заключение приведём образец исходного текста письма, объединяющий все коротенькие примеры данного раздела.

```
\documentclass[12pt,a4paper]{letter}
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
\makelabels
\begin{document}
\address{630090 Новосибирск\\  пр. Науки 99\\ Институт Колдовства}
\signature{К.\,У.~Десник}
\name{Директор Института Колдовства\\
    Академик Хиромантии\\[1in] К.\,У.~Десник}
\date{\today}
\location{офис 304} \telephone{тел: (3832) 39-42-68}
\begin{letter}{Начальнику Департамента Хиромантии\\
    Его сиятельству Кн. В.\,А.~Длиннорукому\\ в собственные руки\\
    С.-Петербург}
\opening{Милостивый государь\\князь Вениамин Алексеевич!}
\begin{flushleft}
    Осмелюсь беспокоить Ваше высокопревосходительство покорнейшей
    просьбой. Соблаговолите распорядиться поставить в офис 013
    Института колдовства персональный компьютер Mac-Plus.
\end{flushleft}
\closing{Вашего высокопревосходительства\\ покорнейший слуга}
\ps{P.S. В крайнем случае сгодится Cray.}
\cc{Президенту\\ Председателю\\ Прокуратору}
\encl{(1) план офиса\ (2) коробка для компьютера}
\end{letter}
\end{document}
```

630090 Новосибирск
пр. Науки 99
Институт Колдовства

28 декабря 2003 г.

Начальнику Департамента Хиромантии
Его сиятельству Кн. В. А. Длиннорукому
в собственные руки
С.-Петербург

Милостивый государь
князь Вениамин Алексеевич!

Осмелюсь беспокоить Ваше высокопревосходительство покорнейшей просьбой. Соблаговолите распорядиться поставить в офис 013 Института колдовства персональный компьютер Mac-Plus.

Вашего высокопревосходительства
покорнейший слуга

К. У. Десник

P.S. В крайнем случае сгодится Cray.

исх.: Президенту
Председателю
Прокуратору

вкл.: (1) план офиса (2) коробка для компьютера

Рис. 15.1. Образец неофициального письма

28 декабря 2003 г.

Начальнику Департамента Хиромантии
Его сиятельству Кн. В. А. Длиннорукому
в собственные руки
С.-Петербург

Милостивый государь
князь Вениамин Алексеевич!

Осмелюсь беспокоить Ваше высокопревосходительство покорнейшей просьбой. Соблаговолите распорядиться поставить в офис 013 Института колдовства персональный компьютер Mac-Plus.

Вашего высокопревосходительства
покорнейший слуга

К. У. Десник

P.S. В крайнем случае сгодится Cray.

исх.: Президенту
Председателю
Прокуратору

вкл.: (1) план офиса (2) коробка для компьютера

офис 304 тел: (3832) 39-42-68

Рис. 15.2. Образец официального письма

Начальнику Департамента Хиромантии
Его сиятельству Кн. В. А. Длиннорукому
в собственные руки
С.-Петербург

Рис. 15.3. Адрес для конверта

Полезно обратить внимание, что основной текст письма мы поместили в тело процедуры `flushleft`, чтобы предотвратить его выравнивание с правой стороны (раздел 5.1). Существует мнение, что письмо с выравниваем текста по обоим краям страницы воспринимается адресатом менее доверительно.

Отформатированный текст неофициального письма показан на рис. 15.1. Если удалить декларацию `\address`, письмо станет официальным. Оно показано на рис. 15.2. На отдельном листе будет напечатан адрес для конверта (рис. 15.3). В неофициальном письме текст, указанный в `\name` (или `\signature`) и `\closing`, смещён к правому краю страницы, а в официальном письме — к левому. В неофициальном письме текст из `\location` и `\telephone` игнорируется, а в официальном письме будет напечатан внизу первой страницы.

Обратим внимание Читателя на текст в аргументе декларации `\name`. Оформление подписи в официальном письме в нашей стране отличается от принятого в некоторых зарубежных странах. В частности, фамилия лица, подписавшего письмо, расшифровывается с правой стороны подписи, а не с левой, где указывается его должность. Класс `letter` запрограммирован на отличный от принятого у нас формат подписи, когда и должность, и фамилия располагаются слева[2].

Если не загружать пакет babel с опцией `russian`, то в списке рассылаемых копий письма вместо слова «копия» появится «cc», а список приложений LaTeX обозначит словом «encl». Замена таких ключевых слов, как обычно, осуществляется путём переопределения команд-логосов. Например:

```
\renewcommand{\enclname}{Приложения}
\renewcommand{\ccname}{Копии}
```

Полный список логосов, имеющих отношение к классу `letter`, приведён ниже.

15.4.1. Параметры настройки

`\ccname` — ключевое слово, которое команда `\cc` печатает перед списком адресатов копий письма.

`\enclname` — ключевое слово, которое команда `\encl` печатает перед списком материалов, прилагаемых к письму.

`\pagename` — ключевое слово, которое LaTeX печатает в верхнем колонтитуле перед номером страницы (раздел 17.1).

[2] Читатель может обратиться к предыдущим изданиям нашей книги [10, 11], где описаны возможные способы изменения формата подписи. В настоящее время в подобном изменении нет насущной необходимости.

`\headtoname` — ключевое слово, которое LaTeX печатает в верхнем колонтитуле перед фамилией адресата (содержащейся в аргументе *adr* процедуры `letter`).

Перечисленные команды-логосы переопределяются посредством `\renewcommand`.

15.5. Класс revtex4

Помимо стандартных классов, рассмотренных выше, многие издательства разработали так называемые специализированные классы документов, дабы учесть особенности своих собственных изданий. В качестве примера подобного класса рассмотрим `revtex4`, разработанный Американским физическим обществом (APS). Этот класс используют многие журналы по физике, как за рубежом, так и в России, но более всего он соответствует стилю журнала «Physical Review». Он распространяется в составе одноимённого пакета REVTeX4, где также имеются шаблон (`template.aps`) и образец (`apssamp.tex`) исходного текста документа, которые можно использовать в качестве стартовой точки при подготовке печатного документа класса `revtex4`.

Цифра 4 в названии класса означает номер версии. Класс `revtex4` не следует использовать с версиями LaTeX'а, выпущенными ранее декабря 1996 года. Он также может конфликтовать с некоторыми пакетами, особенно если их функции уже реализованы в `revtex4`. К таким пакетам относятся multicol, cite и endfloat и float. Классу `revtex4` требуется пакет natbib для работы с библиографическими ссылками (глава 13), однако этот пакет распространяется отдельно[3].

15.5.1. Опции класса revtex4

Выбирая класс `revtex4`, в необязательном аргументе команды `\documentclass` можно указать стиль журнала, куда предполагается направить статью для публикации. Например, стилю журнала «Physical Review A» соответствует опция `pra`. Для журнала «Physical Review B» нужно выбрать опцию `prb`, а журналу «Physical Review Letters» отвечает `prl`. Для обзорных статей в «Reviews of Modern Physics» следует предпочесть `rmp`. Отечественные журналы чаще всего полагаются на опцию `aps`, которая используется по умолчанию. Для предварительных черновых версий документа (препринтов) можно выбрать опцию `preprint` и т. д. Приведённый ниже перечень опций даёт представление о богатстве форматирующих возможностей `revtex4`. В этот перечень не включены стандартные опции, имеющиеся у класса `article`, а также опции, которые имитируют действие некоторых команд или пакетов.

`aps` | `prl` | `pra` | `prb` | `prc` | `prd` | `pre` | `prstab` | `rmp` — выбирает вариант оформления печатного документа. Все опции, кроме последней, предназначены для журналов серии «Physical Review», опция `rmp` соответствует стилю журнала

[3] Как и пакет REVTeX4, он входит в библиотеку MiKTeX.

«Reviews of Modern Physics». По умолчанию используется опция `aps`, подходящая для любых журналов APS.

`preprint` — выбирает стиль, удобный для препринтов; он характеризуется, главным образом, увеличенным межстрочным интервалом и увеличенным размером шрифта (`12pt` вместо `10pt`).

`tightenlines` — используется в совокупности с `preprint`, чтобы уменьшить межстрочный интервал.

`groupedaddress` | `superscriptaddress` | `unsortedaddress` | `runinaddress` — указывает способ группирования фамилий авторов в заголовке документа. Опция `groupedaddress` используется по умолчанию и группирует фамилии по принадлежности к определённому коллективу (обычно по месту работы). Альтернативная опция `superscriptaddress` связывает авторов с коллективом, помечая фамилии верхними индексами; она уместна, если некоторые авторы принадлежат нескольким, но не всем коллективам. Опция `unsortedaddress` действует аналогично `groupedaddress`, но не соединяет авторов из одного коллектива. Наконец, `runinaddress` также действует подобно `groupedaddress`, но располагает фамилии авторов из одного коллектива через запятую в общей последовательности.

`altaffilletter` | `altaffillsymbol` — выбирает вид индекса при обозначении индексами принадлежности авторов к коллективам. Опция `altaffilletter` устанавливает, что индексы печатаются буквами или цифрами. Альтернативная опция `altaffillsymbol` используется по умолчанию и означает, что индексы должны печататься подстрочными символами.

`showpacs` | `noshowpacs` — разрешает или запрещает печать кодов системы классификации PACS[4] на титульной странице документа. По умолчанию используется опция `noshowpacs`, т. е. декларация `\pacs` (см. ниже) игнорируется, даже если она имеется.

`showkeys` | `noshowkeys` — разрешает или запрещает печать ключевых слов на титульной странице документа. По умолчанию используется `noshowkeys`, т. е. декларация `\keywords` (см. ниже) игнорируется, даже если она имеется.

`preprintnumbers` | `nopreprintnumbers` — разрешает или запрещает печать номера публикации (препринта) на титульной странице документа. Совместно с опцией `preprint` по умолчанию используется опция `nopreprintnumbers`, т. е. декларация `\preprint` (см. ниже) игнорируется, даже если она имеется; в ином случае по умолчанию действует опция `nopreprintnumbers`.

`eqsecnum` — устанавливает независимую нумерацию уравнений в каждом новом разделе, начинающимся с команды `\section`.

[4] Она используется зарубежными журналами для сортировки и поиска публикаций по темам.

`bibnotes`|`nobibnotes` — управляет расположением авторских примечаний. Опция `bibnotes` печатает примечания среди списка цитированной литературы. Альтернативная опция `nobibnotes` восстанавливает более привычное размещение примечаний. Опция, действующая по умолчанию, определяется выбранным стилем журнала.

`footinbib`|`nofootinbib` — управляет расположением подстрочных примечаний. Выбор опции, действующей по умолчанию, определяется стилем журнала. Некоторые журналы печатают подстрочные примечания как часть списка цитированной литературы.

`floats`|`endfloats`|`endfloats*` — управляет размещением плавающих объектов. По умолчанию действует опция `floats`, а плавающие рисунки и таблицы размещаются вблизи их положения в исходном тексте. Альтернативная опция `endfloats` перемещает все плавающие объекты в конец документа, аналогично тому, как это делает почти одноимённый пакет endfloat. Опция `endfloats*` к тому же печатает каждый объект на отдельной странице.

`floatfix` — в совокупности с `endfloat` эту опцию следует использовать при получении сообщения об ошибке «`Too many unprocessed floats`».

`balancelastpage`|`nobalancelastpage` — используется при печати в две колонки (когда опция `twocolumn` использована явно или установлена стилем журнала). Опция `balancelastpage` используется по умолчанию и приводит к выравниваю длины колонок на последней странице; альтернативная опция `nobalancelastpage` отменяет выравнивание.

`raggedbottom`|`flushbottom` — используется при печати в две колонки. Опция `flushbottom` используется по умолчанию и приводит к выравниваю нижней границы колонок на каждой странице.

`raggedfooter`|`noraggedfooter` — управляет размещением подстрочных примечаний; опция `noraggedfooter` используется по умолчанию.

`byrevtex` — печатает на полях страниц «Typeset by REVTEX 4».

`galley` — печатает только одну узкую колонку на каждой странице.

15.5.2. Титульная страница документа

Чтобы набрать титульную страницу документа класса `revtex4`, можно использовать команду `\maketitle`, которая имеется в большинстве стандартных классов, в том числе в классе `article`. Однако существует важное отличие во взаимном расположении этой команды и деклараций `\title`, `\author` и `\date`, из которых `\maketitle` получает информацию соответственно о названии, авторах статьи и дате публикации. Все эти декларации должны располагаться перед `\maketitle` в

теле документа, т. е. после `\begin{document}`, тогда как в документах стандартных классов эти декларации могут находиться в преамбуле документа.

Более того, процедура `abstract`, предназначенная для печати аннотации статьи, также должна находиться перед `\maketitle`, тогда как в обычной статье класса `article` аннотация может располагаться как до, так и после `\maketitle` (в первом случае аннотация будет напечатана на отдельной странице).

Ещё одно кардинальное отличие состоит в том, что допускается несколько деклараций `\author`, поскольку в классе `revtex4` одна такая команда предназначена, вообще говоря, для одного автора. Авторы группируются по принадлежности к определённому коллективу или коллективам (организациям). Одна группа состоит из авторов, которые принадлежат к одному набору коллективов. Имя одного автора указывается в аргументе команды

`\author{`*author*`}` (revtex4)

а коллектив (или адрес) указывается в аргументе команды

`\affiliation{`*affiliation*`}` (revtex4)

Группа авторов определяется последовательностью команд `\author`, за которой следует команда `\affiliation`. Она применяется ко всем авторам, т. е. командам `\author`, для которых ещё не определена группа (адрес).

Например, если Bugs Bunny и Roger Rabbit оба относятся к Looney Tune Studios, а Mickey Mouse принадлежит Disney World, разметка должна быть такой:

```
\author{Bugs Bunny}
\author{Roger Rabbit}
\affiliation{Looney Tune Studios}
\author{Mickey Mouse}
\affiliation{Disney World}
```

Bugs Bunny and Roger Rabbit
Looney Tune Studios

Mickey Mouse
Disney World

Справа показано, как выглядит список авторов, если в `\documentclass` выбрана опции `groupedaddress`. Поскольку она используется по умолчанию во всех журналах APS, её можно не указывать явно.

Следующий пример показывает, что получается при выборе альтернативной опции `superscriptaddress`. Обратите внимание, что Roger Rabbit приписан к обоим коллективам:

```
\author{Bugs Bunny}
\author{Roger Rabbit}
\affiliation{Looney Tune Studios}
\affiliation{Disney World}
\author{Mickey Mouse}
\affiliation{Disney World}
```

Bugs Bunny,[1] Roger Rabbit,[1,2] and Mickey Mouse[2]

[1]*Looney Tune Studios*
[2]*Disney World*

Класс `revtex4` без подсказки расставляет знаки препинания в списке авторов и выбирает нужный шрифт. Только имена авторов и названия (или адреса) авторских коллективов должны быть заданы в аргументах соответствующих команд.

Опция groupedaddress сортирует авторов по соответствующим группам, если \affiliation имеется для каждого автора. В следующем примере будет получен тот же результат, что и в первый раз, хотя Roger Rabbit в исходном тексте поставлен после Mickey Mouse:

```
\author{Bugs Bunny}
\affiliation{Looney Tune Studios}
\author{Mickey Mouse}
\affiliation{Disney World}
\author{Roger Rabbit}
\affiliation{Looney Tune Studios}
```

Bugs Bunny and Roger Rabbit
Looney Tune Studios

Mickey Mouse
Disney World

Чтобы избежать перестановок фамилий, используйте опцию unsortedaddress вместо groupedaddress. Лучше всего перечислить фамилии авторов в том порядке, в котором они должны быть напечатаны, и указать \affiliations для группы авторов, а не для каждого в отдельности.

Если используется опция superscriptaddress, коллективы нумеруются в том порядке, в каком они даны в исходном тексте. Это означает, что очерёдность коллективов определяется очерёдностью авторов. Иной порядок может быть получен, если вставить список всех \affiliation до первой команды \author, а затем указать перечень \affiliation для каждого автора вслед за соответствующей командой \author. Если автор не принадлежит ни одному коллективу, можно использовать команду

`\noaffiliation` (revtex4)

вместо \affiliation.

Названия коллабораций (объединений нескольких коллективов) следует вводить в аргумент декларации

`\collaboration{`*collaboration*`}` (revtex4)

Она более всего похожа на команду \author, но может быть использована только в совокупности с опцией superscriptaddress. Команда \collaboration должна находиться в конце списка авторов после команд \author. Название коллаборации печатается в скобках перед списком коллективов. Поскольку коллаборации обычно не имеют адреса, нужно вслед за \collaboration размещать команду \noaffiliation.

На первой странице документа не следует использовать команду \footnote. Если необходимо сообщить дополнительную информацию об авторе, коллективе или коллаборации, нужно использовать команды

```
\email[text]{e-mail}
\homepage[text]{url}
\altaffiliation[text]{affiliation}
\thanks{text}
```
(revtex4)

Первые три из них указывают соответственно адрес электронной почты, адрес сайта в интернете, альтернативный коллектив и имеют необязательный аргумент *text*, который может содержать дополнительную информацию. Эта информация будет напечатана перед текстом из обязательного аргумента взамен печатаемого по умолчанию. Например, если необязательный аргумент пропущен у команды `\email`, будет напечатано «Electronic address:». Четвертую команду `\thanks` следует использовать, только если первые три не соответствуют содержанию дополнительной информации; она сохранена, главным образом, для совместимости со стандартными классами документов.

Любому автору можно приписать более чем по одному экземпляру любой из четырёх команд, но в отличие от `\affiliation`, каждая такая команда действует только на одну команду `\author`, непосредственно предшествующую ей. Обычно дополнительная информация размещается в подстрочных примечаниях:

```
\author{Bugs Bunny}
\email[Пишите мне на адрес ]{bugs@looney.com}
\homepage{http://looney.com/}
\altaffiliation[место работы: ]
   {Warner Brothers}
\affiliation{Looney Tunes}
```

[1]Пишите мне на адрес: `bugs@looney.com`, Url: `http://looney.com/`, место работы: Warner Brothers.

Примечания к командам `\collaboration`, `\affiliation` или даже `\title` также можно сделать через команды `\thanks`, `\email`, `\homepage`.

Дубликаты примечаний будут объединены, но это верно только при условии, что примечания полностью идентичны; при этом даже порядок команд с дополнительной информацией имеет значение. Таким образом, два автора могут делить единое подстрочное примечание с указанием группы адресов электронной почты.

Дубликаты команды `\affiliation` также могут присутствовать в исходном тексте. Однако дополнительная информация к `\affiliation` должна быть дана только к первому экземпляру команд `\affiliation` с идентичными аргументами.

Размещение дополнительной информации в подстрочных примечаниях варьируется от журнала к журналу. Так, опция `prb` переносит все примечания с первой страницы в начало списка цитированной литературы, но другие журналы размещают эту информацию всё же на первой странице. Можно изменить размещение, используемое по умолчанию, явно указав опцию `bibnotes` в `\documentclass` (переносит примечание в начало списка литературы) или `nobibnotes` (оставляет их на первой странице).

Некоторые авторы в своём имени сначала пишут фамилию и лишь затем собственно имя или инициалы. Неопределённость также возникает в том случае, когда фамилия или имя состоят из нескольких слов. Чтобы снять возможные недоразумения, особенно нежелательные для поисковых машин, работающих в интернете, предлагается использовать команды

`\surname{`*surname*`}` `\firstname{`*firstname*`}`	(revtex4)

в аргументе команды `\author`. Первая из них должна содержать фамилию, вторая — имя. Наличие команды `\surname` делает избыточным, хотя и не запрещает, использование команды `\firstname`. Эти две команды внешне ничего не меняют:

`\author{Andrew \surname{Lloyd Weber}}`	Andrew Lloyd Weber
`\author{\surname{Mao} Tse-Tung}`	Mao Tse-Tung

Текст аннотации статьи нужно поместить в тело процедуры `abstract`:

`\begin{abstract} ... \end{abstract}` (revtex4)

Однако он будет напечатан только командой `\maketitle`. Следовательно, процедура `abstract` должна предшествовать этой команде.

Декларации

`\pacs{`*codes*`}`
`\keywords{`*keywords*`}` (revtex4)

объявляют код документа по классификации PACS и ключевые слова:

```
\pacs{23.23.+x, 56.65.Dy}
\keywords{nuclear form; yrast level}
```

Аргументы этих команд *codes* и *keywords* реально печатает команда `\maketitle`, как правило, после аннотации, но лишь при условии, что в `\documentclass` явно указаны соответственно опции `showpacs` и `showkeys`.

Номер издания по классификации организации, которая публикует статью, объявляет декларация

`\preprint{...}` (revtex4)

Номер будет напечатан командой `\maketitle` в верхнем левом углу первой страницы. Можно использовать несколько деклараций `\preprint`, но из-за ограниченности свободного места будут напечатаны не более трёх номеров.

Наконец, печать заголовка, списка авторов, аннотации, кодов PACS, ключевых слов, номера препринта, производит команда

`\maketitle` (revtex4)

Она должна располагаться после деклараций и процедур, перечисленных выше. Хотя её синтаксис не отличается от одноимённой команды в стандартных классах документов, нужно признать, что по сути это совсем другая команда. Фактически она печатает всю первую страницу документа.

15.5.3. Особенности класса revtex4

Журналы «Physical Review» печатаются в две колонки. Чтобы получить представление, как будет выглядеть статья в двухколоночном формате, нужно указать опцию twocolumn в \documentclass. При этом возникает потребность часть материала напечатать в одну колонку. Стандартные классы для этой цели предлагают команду \onecolumn, но она начинает печать в одну колонку со следующей страницы. Класс revtex4 предлагает решение, более соответствующее особенностям научных статей, когда во всю ширину страницы, как правило, нужно напечатать 1–2 сложных уравнения. Такие уравнения достаточно разместить в теле процедуры

```
\begin{widetext} ... \end{widetext}
```
(revtex4)

Эта процедура выравнивает предшествующий ей текст в две короткие колонки равной высоты и рисует две горизонтальные черты на всю ширину страницы, между которыми размещает своё тело, после чего вновь восстанавливает печать в две колонки. Применение процедуры widetext должно быть ограничено одним-двумя длинными уравнениями, между которыми допускается незначительное вкрапление текста. Процедура widetext нечего не меняет, если документ компилируется с опцией preprint, поскольку в этом случае весь текст печатается в одну колонку.

Широкие рисунки и широкие таблицы следует печатать с использованием процедур figure* и table*, которые размещают плавающие объекты, используют всю ширину страницы, как в одно-, так и в двухколоночном формате. Не следует размещать figure и table в теле процедуры widetext.

В журналах APS принято отделять таблицы двойной чертой от окружающего текста. Чтобы получить такое оформление, нужно поместить таблицу в тело процедуры

```
\begin{ruledtabular} ... \end{ruledtabular}
```
(revtex4)

как показывает следующий пример:

```
\begin{table}
    \caption{...}\label{...}
    \begin{ruledtabular}
        \begin{tabular}
            ...
        \end{tabular}
    \end{ruledtabular}
\end{table}
```

Если ширина рисунка или таблицы превышает ширину страницы, её можно повернуть на 90°. Для этого нужно поместить процедуру figure или table, формирующую плавающий объект, в тело процедуры

`\begin{turnpage} ... \end{turnpage}` (revtex4)

предварительно загрузив пакет graphics или graphicx:

```
\documentclass[...]{revtex4}
\usepackage{graphicx}
    ...
\begin{turnpage}
    \begin{figure}
        ...
    \end{figure}
\end{turnpage}
```

При использовании опции `endfloats` или `endfloats*` с классом `revtex4` все плавающие объекты задерживаются до конца документа. Задержанные объекты можно напечатать в месте, обозначенном командами

`\printtables`	`\printtables*`
`\printfigures`	`\printfigures*`

(revtex4)

`*`-форма этих команд начинает печать с новой страницы. Тот же эффект достигается даже обычной формой команд, если класс `revtex4` выбран с опцией `endfloats*`; при этом каждый объект размещается на отдельной странице. Без опций `endfloats` или `endfloats*` эти команды игнорируются, поэтому их можно не удалять. Если же команды пропущены, но опции выбраны, плавающие объекты будут напечатаны в самом конце документа.

В настоящее время зарубежные журналы не требуют, чтобы рисунки и таблицы обязательно были размещены после основного текста; и редакторы, и рецензенты предпочитают иметь иллюстрации «по месту». Напротив, некоторые редакции отечественных журналов продолжают жить по устаревшим правилам.

Наконец, процедура

`\begin{acknowlegments} ... \end{acknowlegments}` (revtex4)

печатает раздел «Acknowledgments» («Благодарности»), где принято указывать источники финансирования публикуемой работы и благодарить коллег. Разные журналы используют свои способы оформления этого раздела, но все отличия будут автоматически учтены.

Не позволим дилетантам вернуть свои деньги.
Девиз редактора сборника решений судов Канады

Глава 16

Шрифты для профессионалов

Некоторое время после своего изобретения LaTeX мог работать только со шрифтами METAFONT [3]. Выбор шрифтов METAFONT и по сей день невелик, поэтому документы LaTeX имели легко узнаваемый вид из-за преимущественного использования шрифтов семейства Computer Modern, разработанного Д. Кнутом.

Современный LaTeX обеспечивает доступ к шрифтам любых типов, которые используются в компьютерных системах. Он избавился от существовавшего ранее ограничения на число подключаемых шрифтов. Однако процедура подключения шрифтов, разработанных для иных компьютерных приложений, не очень-то проста. Рядовой пользователь должен использовать пакеты шрифтов, адаптированные для LaTeX'а специалистами. Процедура адаптации заключается в генерации метрических файлов шрифтов (с расширением `tfm`) и файлов определения шрифтов (с расширением `fd`). Эта тема, как принято говорить в подобных случаях, выходит за рамки данной книги. Рядовому пользователю достаточно знать, как установить существующий пакет шрифтов, а эта процедура мало отличается от установки любого другого пакета системы LaTeX, описанной в разделе 3.3.7. Однако если обычный пакет состоит из файлов с расширением `sty`, то шрифтовый пакет включает ещё много файлов других типов.

Как только пакет установлен, достаточно загрузить его в редактируемый документ с помощью `\usepackage`. Например, перед компиляцией данной главы мы загрузили пакет TimesC:

```
\usepackage[math]{TimesC}
```

Он заменил шрифты CM Roman, которыми набраны другие главы книги, на шрифты Times New Roman. В этой главе мы расскажем ещё о нескольких пакетах, предназначенных для работы со шрифтами, которые распространяются либо бесплатно, либо в составе других продуктов, так что их приобретение не требует дополнительных затрат.

PSNFSS, PSNFSSx — пакеты коллекции PSNFSS используются для подключения набора 35 стандартных шрифтов PostScript фирмы Adobe, содержащих только буквы латинского алфавита. Бесплатный клон этих шрифтов распространяется в составе программы Ghostscript. Коллекция PSNFSSx является усовершенствованной версией пакетов PSNFSS (раздел 16.8).

pxfonts, txfonts — пакеты pxfonts и txfonts подключают шрифты PostScript соответственно в гарнитурах Palatino и Times как в тексте, так и в математических фор-

мулах, создавая сбалансированный печатный документ. Необходимые шрифты распространяются в составе бесплатной программы Ghostscript, но в них отсутствуют русские буквы (раздел 16.9).

pscyr — пакеты коллекции pscyr подключают некоторые свободно распространяемые русские PostScript шрифты (раздел 16.10).

FontsC — пакеты коллекции FontsC подключают шрифты TrueType и OpenType, входящие в состав русифицированных версий операционной системы Windows, а также некоторые свободно распространяемые русские PostScript шрифты (раздел 16.10).

Читатель может обратиться непосредственно к указанным разделам, если всё, что ему нужно,— подменить шрифты сразу во всём документе. Однако мы бы не советовали пропускать следующие два раздела, посвящённые характеристикам шрифтов. В разделе 16.3 мы расскажем о командах низшего уровня, которые позволяют делать то, что неподвластно командам так называемого пользовательского уровня, рассмотренным в разделе 16.4.

16.1. Типы шрифтов

16.1.1. Шрифты METAFONT

Д. Кнут назвал созданную им коллекцию из 31 шрифта семейством Computer Modern, хотя в традиционном понимании это скорее метасемейство, т. е. набор семейств шрифтов. METAFONT позволяет получить из одного исходного описания множество совершенно различных шрифтов. Новые шрифты получаются варьированием нескольких входных параметров. Прямые, наклонные, курсивные, контурные (пустотелые), жирные и тонкие, с засечками (серифами) и рубленые — все получаются из одного «исходника», которым служит небольшая программа, записанная в файле с расширением `mf`, которая рисует контуры буквы или иного символа.

В своей основе шрифты METAFONT являются векторными. Однако процесс растеризации, т. е. преобразования в пикселы, занимал слишком много времени даже на самых мощных компьютерах, имевшихся на момент изобретения системы METAFONT. Поэтому шрифты приходилось генерировать заранее и хранить в растровом виде на диске компьютера в виде файлов с расширением `pk`. Чем выше разрешение принтера, тем бóльший объём памяти необходим для хранения файлов с растровыми шрифтами. Для принтера каждого типа был необходим отдельный комплект `pk`-шрифтов. Авторы этой книги ещё помнят то время, когда процесс генерации такого комплекта занимал несколько часов.

С ростом производительности компьютеров необходимость в предварительной генерации растровых шрифтов отпала сама собой и появилась возможность генерировать растр «на лету», т. е. по мере возникновения потребности в шрифтах с определённым размером и разрешением. Первой программой, где была реализована генерация

pk-шрифтов «на лету», была многократно упоминавшаяся программа dvips, а произошло это в 1994 году. Однако вновь сгенерированные растровые шрифты до сих пор записываются на жёсткий диск компьютера, дабы сэкономить время в следующий раз.

16.1.2. Шрифты PostScript

Во второй половине 80-х годов получили распространение векторные шрифты PostScript. Процесс растрирования для них несколько проще и поэтому с самого начала производился «на лету» в момент вывода документа на печатное устройство. Первые шрифты PostScript были встроены в принтеры LaserWriter, поэтому до сих пор можно встретить упоминание о 14 или 35 стандартных шрифтах фирмы Abobe.

Программа dvips позволяет использовать шрифты PostScript наравне с растровыми. Она была написана для преобразования документа DVI, созданного компилятором latex, в PostScript-документ (рис. 1.6 на стр. 44). При этом шрифты PostScript можно либо внедрить в полученный ps-файл, как это происходит с pk-шрифтами, либо только вставить ссылку на эти шрифты. В последнем случае шрифты должны быть доступны в момент вывода документа на печать или экран монитора.

Современные DVI-обозреватели также могут работать со шрифтами PostScript. Некоторые обозреватели используют эти шрифты, что называется, напрямую. Другие же «на лету» генерируют из них pk-шрифты; именно так действует YAP из библиотеки MiKTeX.

За пределами России PostScript-шрифты де-факто стали стандартом для большинства реализаций LaTeX'а, вытеснив pk-шрифты. Этому содействовало наличие множества хороших коммерческих шрифтов. Имеется также PostScript-версия шрифтов семейства Computer Modern. Однако среди этих шрифтов практически отсутствуют такие, где были бы русские буквы. Лишь фирма ParaType[1] предоставила русским пользователям LaTeX'а в бесплатное пользование шрифты гарнитуры «Литературная». PostScript-версия mf-шрифтов семейства LH, где имеются буквы кириллицы, появилась в 2001 году в составе комплекта PostScript-шрифтов CM-Super, разработанных Владимиром Воловичем.

Шрифты CM-Super относятся к типу композитных шрифтов, которые по классификации фирмы Adobe обозначаются как Type 0. Каждый шрифт Type 0 может содержать тысячи символов, упорядоченных в виде дерева, листья которого содержат шрифт Type 1. Шрифт Type 1 написан на языке PostScript и организован в виде набора именованных процедур, рисующих контуры изображения отдельного символа. Когда запрашивается символ с определённым кодом, программа-растеризатор сначала просматривает вектор кодировки, который устанавливает соответствие кода имени процедуры. Затем найденная процедура рисует растровое изображение для текущего разрешения выходного устройства.

Шрифты pk, будучи внедрёнными в ps-файл, относятся к Type 3. Они внедряются в виде растра, поэтому качество документа может деградировать, если он будет на-

[1] Фирма ParaGraph International основана в 1989 году в Москве. В начале 1998 года отдел шрифтов этой компании выделился в самостоятельную компанию ParaType. Адрес сайта компании ParaType: http://www.paratype.com/.

печатан на устройстве, разрешение которого отличается от заказанного при создании документа.

16.1.3. Шрифты TrueType

Фирма Apple приступила к разработке технологии, которая сейчас называется TrueType, в конце 1987 года. В августе 1989 года работа была завершена, а месяцем позже Apple и Microsoft заключили стратегический альянс против компании Adobe, пытавшейся навязать технологию шрифтов PostScript производителям операционных систем. В марте 1991 года Apple выпустила TrueType в свет в виде 80-килобайтного приложения для System 6.0 и шрифтов Times Roman, Helvetica и Courier. С тех пор шрифты TrueType встроены в операционную систему Mac OS для компьютеров Macintosh. Фирма Microsoft встроила технологию TrueType в операционную систему Windows версии 3.1 в начале 1992 года, снабдив её великолепными шрифтами Times New Roman, Arial и Courier. Однако в Windows 3.1 технология TrueType была реализована в усечённом виде, поскольку эта операционная система была 16-разрядной (в отличие от Mac OS, которая уже тогда была 32-разрядной). Только в августе 1995 года с появлением 32-разрядной Windows 95 фирма Microsoft внедрила в свои продукты сглаживание контуров букв (anti-aliasing), что существенно улучшило качество отображения текста на устройствах с низким разрешением (на экране монитора).

В 1996 году Adobe и Microsoft договорились о создании шрифтов OpenType — нового поколения TrueType. Они могут содержать более 65 000 тысяч символов, тогда как в Type 1 можно одновременно использовать не более 256 символов. Шрифты OpenType поставляются в составе операционных систем Windows начиная с Windows 2000.

Шрифты TrueType на удивление мало используются в системе LaTeX. Тому есть простое объяснение. Во-первых, они появились позже, чем шрифты PostScript. Во-вторых, шрифты TrueType невозможно внедрить в документ PostScript без потери качества, а без внедрённых шрифтов документ может неправильно отображаться на другом компьютере. Шрифты TrueType (и OpenType) успешно внедряются в документы PDF, но эта возможность, как и формат PDF, появилась сравнительно недавно. Современные DVI-обозреватели обеспечивают работу со шрифтами TrueType, как правило генерируя на лету pk-шрифты из исходных векторных изображений литер.

Использование шрифтов TrueType практически решает проблему дефицита хороших шрифтов для русских пользователей LaTeX'а. Пакеты из коллекции FontsC, описанные в разделе 16.10, позволяют подключить наиболее распространённые шрифты, имеющиеся в составе операционной системы Windows.

16.2. Характеристики шрифтов

Сначала мы кратко опишем основные характеристики шрифтов, а в следующем разделе расскажем о так называемых командах низкого уровня, которые используются для загрузки шрифтов, но не были описаны в главе 1. С их помощью опытный пользователь может подключить шрифты, не прибегая к загрузке пакетов посредством

\usepackage. Большинство шрифтовых пакетов состоят из одной, максимум трёх подобных команд, а поскольку комбинаций выбора шрифтов может быть огромное множество, просто нецелесообразно писать такие пакеты на все мыслимые и немыслимые случаи жизни.

Шрифты характеризуются несколькими параметрами. Мы ограничимся их кратким обзором, отсылая любознательного Читателя к специальной литературе [14, 21].

16.2.1. Кодировка

Кодировка устанавливает соответствие между кодом символа и литерой, т. е. его изображением. Читателю, возможно, приходилось перекодировать текст при переносе его с персонального компьютера, где в операционной системе MS DOS используется кодовая страница cp866, на компьютер под управлением Unix, где используется кодировка KOI8. В русифицированных версиях Windows применяется кодировка ANSI 1251.

LaTeX вводит собственную *схему кодирования*, которую мы будем называть *внутренней* кодировкой, чтобы отличить её от *внешней* кодировки, т. е. *кодовой страницы*, используемой операционной системой. Примером внутренней кодировки являются T1, T2A, OML. Как правило, внешняя кодировка должна быть указана в исходном тексте документа LaTeX; это особенно важно для языков, подобных русскому, которые исторически имели множество вариантов кодировки. Внутреннюю кодировку, используемую по умолчанию, выбирает пакет babel в зависимости от указанного набора языков, используемых в документе.

Есть ещё кодировка внутри файлов шрифтов, но она меньше всего должна заботить рядового пользователя LaTeX'а. Шрифты METAFONT могут содержать не более 256 символов; соответственно кодом служит число от 0 до 255. Шрифты OpenType могут содержать десятки тысяч символов и, соответственно, имеют более сложную схему кодирования. Задача разработчика шрифтового пакета состоит как раз в том, чтобы избавить пользователя от лишней головной боли.

16.2.2. Гарнитура

Гарнитура определяет художественное решение, выделяющее шрифт среди других. Например, в данной главе используется гарнитура Times New Roman. Есть гарнитуры Академическая, Литературная, Garamond, Bookman Old Style и много других. В метасемействе шрифтов Computer Modern можно выделить гарнитуры CM Roman, CM Sans Serif, CM Typewriter. По терминологии разработчиков системы LaTeX гарнитура образует семейство *family*. Гарнитуры различаются по контрастности символов и наличию засечек.

Контрастность

Контрастность определяется отношением толщины горизонтальных и вертикальных штрихов символов. Гарнитура Times более контрастна, чем гарнитура CM Roman, которая в свою очередь контрастнее гарнитуры Courier.

Засечки

Чаще всего используют шрифты с серифами. Серифами, или засечками, называют небольшие черточки на концах линий, образующих тот или иной символ. Шрифты Computer Modern Sans Serif не имеют засечек. Такие шрифты называют рублеными. Гарнитуры с засечками иногда называют романскими (roman), поскольку засечки оставлял резец, когда заглавные латинские буквы вырезали на памятниках Римской Империи.

16.2.3. Насыщенность и пропорциональность

В группе шрифтов, принадлежащей к одной гарнитуре, выделяют шрифты с различной насыщенностью и пропорциональностью.

Насыщенность — это относительная толщина штрихов символов. В следующей последовательности каждый второй символ имеет бо́льшую насыщенность, чем предшествующий ему: А**А**, Б**Б**, *В**В***. В руководствах по издательским системам встречаются такие наименования насыщенности (в порядке её увеличения):

Ultra Light, Thin, Extra Light, Light	*светлые* и *сверхсветлые* шрифты
Book, Regular, Plain, Normal, Medium	*нормальные* шрифты
Demi, Demi Bold, Semi Bold, Bold	*полужирные* шрифты
Extra Bold, Heavy, HeavyFace, Black, Flat, Extra Black, Ultra Black, Obese	*жирные* и *сверхжирные* шрифты

Пропорциональность шрифта связывают с шириной букв. Для точного определения выбирается буква М и вычисляется отношение её ширины и высоты. Различают узкие, нормальные и широкие гарнитуры.

Very Condensed, Condensed	*сверхузкие* и *узкие* шрифты
Normal	*нормальные* шрифты
Expanded, Very Expanded	*широкие* и *сверхширокие* шрифты

Кроме пропорциональных имеются шрифты с фиксированной шириной символов — *моноширинные* шрифты; их называют также *машинописными*. Они применяются при наборе листингов компьютерных программ, а также в печатных машинках. Уместно использование моноширинных шрифтов в частных письмах. Примером шрифтов с фиксированной шириной символов являются гарнитуры `Typewriter` и `Courier`.

В LaTeX'е насыщенность и пропорциональность шрифта характеризует его серия series.

16.2.4. Начертание

Почти все гарнитуры имеют так называемое прямое (upright) *начертание*[2]. Иногда начертание upright не совсем точно называют roman. Большинство книг, в том числе и

[2] Другой термин — shape (форма шрифта).

наша, набраны прямым шрифтом, а *курсивный* (italic) и *наклонный*[3] (slanted, oblique) шрифты используются для смыслового выделения текста. Обычно курсив одновременно является наклонным, но это вовсе не обязательно. Курсивный шрифт может быть и прямым (upright italic). Ещё одно распространённое начертание — Капитель (small caps). В нём строчные буквы похожи на прописные (заглавные), но меньше их. Отдельная статья — *рукописные* шрифты. Обычно рукописные гарнитуры поставляются только в одном начертании, которое формально классифицируют как курсивное.

16.2.5. Кегль

Размер шрифта, то есть его *кегль*, традиционно измеряют в пунктах `pt`. Кегль у компьютерных шрифтов не является абсолютной мерой какой-то определённой характеристики, скорее это некоторое значение, выбранное разработчиком шрифта для ориентировки пользователей. Различие особенно заметно, когда гарнитуры от разных производителей используются в одном документе и набраны одним кеглем: Times New Roman, Академическая, Литературная, Garamond, Bookman Old Style. Некоторые шрифтовые пакеты позволяют нивелировать видимые различия, вводя масштабные коэффициенты.

16.2.6. Качество шрифтов

Существует ещё ряд характеристик, которые отражают технологическое совершенство шрифтов.

В некоторых словах имеются комбинации символов, расстояние между которыми кажется непропорционально большим, как расстояние между буквами «Г» и «д» в слове «Где». Этот дефект невозможно устранить, уменьшив ширину буквы Г, так как она будет сливаться с соседними буквами в других комбинациях. Решение даёт специальный метод изменения расстояния между символами, называемый *кернингом*. Как правило, говорят о пáрном кернинге, который изменяет расстояние между символами, входящими в определённые пары. «Г» и «д» составляют одну из таких пар. С кернингом слово «Где» выглядит значительно лучше. Количество пар кернинга зависит от шрифта. Практика показывает, что нескольких сотен пар достаточно для качественного воспроизведения текста.

Другой способ улучшения внешнего вида текста — изменение расстояния между символами в зависимости от кегля (размера) шрифта. Этот метод называется *трекингом*. При наборе основного текста обычно используются шрифты небольшого кегля (как правило, от 10 до 12 пунктов), и для улучшения читаемости расстояния между символами немного увеличиваются. С увеличением размера шрифта относительное расстояние между символами постепенно уменьшается. Это повышает компактность текста и позволяет воспринимать его как цельное графическое изображение. Трекинг

---

[3] В гарнитуре Times New Roman, которая используется в данной главе, наклонное начертание отсутствует. Наклонный шрифт может быть получен путём трансформации прямого шрифта. Однако некоторые программы, в том числе `pdflatex`, не поддерживают трансформацию наклона шрифтов TrueType, а внедрение трансформированного шрифта в документ PDF в растровом виде сопряжено с потерей качества.

шрифтов METAFONT обеспечивается наличием шрифтов, специально спроектированных для наиболее часто используемых размеров (см. ниже). В других шрифтах трекинг вычисляется по специальным формулам для любого размера.

Шрифты PostScript и TrueType содержат дополнительную информацию, называемую *хинтами*, которая позволяет улучшить растровое изображение при значительном уменьшении или увеличении размера шрифта. Особенно полезны хинты при печати документов на устройствах с малым разрешением, поскольку округление до целых значений координат контуров букв приводит к исчезновению одних линий и утолщению других. Хинты препятствуют этому, уравнивая толщину линий и сохраняя тем самым эстетическую привлекательность литер.

16.3. NFSS, или Ортогональная схема выбора шрифтов

Произвольный шрифт в LaTeX'е полностью характеризуется набором из пяти атрибутов: внутренняя кодировка ENC, гарнитура *family*, насыщенность *series*, начертание *shape* и размер *size*. На восприятие текста влияет также расстояние между строками, которое можно изменять, задавая расстояние *baselineskip* в единицах длины или вводя масштабный коэффициент *baselinestretch*. Любой атрибут можно задавать независимо от других, получая в результате множество разных вариантов. Такая схема выбора шрифтов при своём появлении получила название New Font Selection Scheme (NFSS), т. е. *новая схема выбора шрифтов*. Поскольку она давно перестала быть новой, её иногда называют ортогональной, чтобы подчеркнуть её основное отличие от схемы, использовавшейся в предыдущей версии LaTeX 2.09, где невозможно было изменить насыщенность шрифта, не изменив его начертание.

Отдельные характеристики текущего шрифта изменяются при помощи деклараций

```
\fontencoding{ENC}
\fontfamily{family}
\fontseries{series}
\fontshape{shape}
\fontsize{size}{baselineskip}
\linespread{baselinestretch}
```

Каждая из них устанавливает один из атрибутов; исключение составляет декларация `\fontsize`, которая задаёт одновременно два параметра. Поскольку параметры *baselineskip* и *baselinestretch* характеризуют не столько шрифты, сколько макет полосы набора, имеется и другой способ их настройки (раздел 17.2).

Перечисленные декларации только подготавливают переключение шрифта. Так как зачастую нужно изменять несколько атрибутов одновременно, было бы неразумно загружать шрифты после указания отдельного атрибута. Реальное переключение шрифта производит декларация

```
\selectfont
```

Таблица 16.1

Стандартные кодировки

ENC	Кодировка	Примечание
OT1	TEX text	кодировка Д. Кнута
T1	TEX extended text	EC кодировка (Корк)
T2A	T2AEncoding	латиница и кириллица
T2B	T2BEncoding	латиница и кириллица
T2C	T2CEncoding	латиница и кириллица
Lxx	A local encoding	локальная кодировка
U	Unknown	неизвестная кодировка
OML	TEX math italic	математический курсив
OMS	TEX math symbols	математические символы
OMX	TEX math large symbols	математические большие символы

которая должна следовать за перечисленными декларациями без всякого промежутка (допускаются только пробелы). Набор изменяемых атрибутов при этом может быть любым. Например, можно изменить гарнитуру и насыщенность шрифта или можно изменить одно начертание. Важно только, чтобы между декларированием атрибутов и `\selectfont` не было иного текста.

Декларация

`\usefont{ENC}{family}{series}{shape}`

есть сокращённый вариант набора деклараций `\font`... с последующим переключателем `\selectfont`, устанавливающий четыре из пяти атрибутов (кроме размера).

В обычном документе, как правило, ни одну из перечисленных выше деклараций не используют. К их помощи приходится прибегать для получения шрифтов с необычным сочетанием атрибутов, например прямого курсива.

```
...например
{\usefont{T2A}{cmr}{m}{ui}прямого курсива}.
```

...например прямого курсива.

Типичные ситуации обслуживаются более простыми командами, известными Читателю из первой главы. Мы ещё вернёмся к ним в разделе 16.4, а сейчас расскажем, какие значения могут принимать аргументы ENC, *family*, series, shape, size и baselineskip.

16.3.1. Стандартные кодировки

Существуют два основных класса кодировок: текстовые и математические. Первые преимущественно используются в обычном тексте: вторые — только в математических формулах. Аббревиатура текстовых кодировок *family* начинается с букв `T` или `OT`. Аббревиатуры математических кодировок должны начинаться с букв `M` или `OM`. Список стандартных кодировок приведен в табл. 16.1. Если не загружен пакет babel, то в текстовом режиме по умолчанию используется кодировка `OT1`. Чтобы выбрать кодировку `T1`, во входной файл можно вставить

```
\fontencoding{T1}\selectfont
```

Попытка тем же способом включить кодировку T2A закончится крахом, если не загружен пакет babel (с опцией russian, bulgarian, или ukrainian), поскольку по умолчанию загружены только кодировки OT1, T1, OML, OMS, OMX:

```
! LaTeX Error: Encoding scheme ‘T2A’ unknown⁴.
 ...

l.15 \fontencoding{T2A}
                       \selectfont
?
```

Любую кодировку можно подготовить к использованию, загрузив пакет fontenc, который описан в разделе 16.5. Пакет babel с опцией russian, который всегда следует загружать при подготовке документов на русском языке, избавляет от заботы о выборе требуемой кодировки, производя все необходимые приготовления. По умолчанию он использует кодировку T2A; её отличие от T2B и T2C незаметно для человека, пишущего по-русски.

Переключение математических кодировок посредством \fontencoding было бы совершенно абсурдной затеей, так как почти любое математическое выражение содержит все три имеющиеся на данный момент математические кодировки OML, OMS и OMX. В математическом режиме компилятор производит переключение кодировок без внешних указаний, которые могли бы только повредить.

Обсуждая способы набора текста в главе 4, мы отмечали, что не все команды одинаково доступны для каждой кодировки. Тут нечему удивляться, если учесть различия в алфавитах и полиграфических традициях различных языков. Теперь добавим несколько слов в целом о каждой из наиболее важных текстовых кодировок.

Кодировка OT1

Внутренняя кодировка, введённая создателем TEX’а Д. Кнутом, содержит всего 128 символов. Она называется TEX text encoding и в LATEX 2ε имеет обозначение OT1, причём буква «O», означающая «old» или «obsolete», подчеркивает, что эта кодировка считается устаревшей. В основном она содержит буквы латинского алфавита, занимая первую (верхнюю) половину 256-местной таблицы C.1 на стр. 451 с кодами от 0 до 127 (или от "00 до "7F в шестнадцатиричном исчислении).

Первые версии LATEX’а не могли работать со шрифтами, содержащими бóльшее количество символов. Позднее, когда максимальное число символов в шрифте было увеличено до 256, во вторую (нижнюю) половину таблицы стали добавлять буквы из других алфавитов. Так возникли более современные кодировки, которые мы рассмотрим далее.

[4] Ошибка LATEX’а: Кодировка T2A не известна.

Кодировки T1 и TS1

В кодировке OT1 машинописные и капительные шрифты имеют некоторые отличия от остальных шрифтов. Например, у машинописных шрифтов отсутствует большая часть лигатур, как видно из сравнения таблиц C.1 и C.2.

Чтобы исключить подобные различия, в 1990 году на совещании в ирландском местечке Корк (Cork) была принята кодировка, содержащая 256 символов. Она называется TEX extended text encoding[5], а в LATEX 2ε обозначается T1. Эта кодировка содержит буквы национальных алфавитов почти всех европейских языков, кроме кириллицы (табл. C.3). Главной целью её введения было обеспечение корректного переноса слов, содержащих буквы с диакритическими знаками. Во имя той же цели часть символов была выделена в отдельную кодировку TS1 (табл. C.4). Кодировка TS1 не предназначена для самостоятельного использования — в ней просто нет обычных букв. Чтобы получить доступ к символам, содержащимся в ней, лучше всего загрузить пакет textcomp:

```
\usepackage{textcomp}
```

Шрифты для кодировок T1 и TS1 разработаны Йоргом Кнаппеном (Knappen, Jörg) и называются European Computer Modern или Extented Computer Modern.

Кодировки T2A, T2B, T2C

Символы кириллицы содержатся в кодировках T2A, T2B, T2C, разработанных русским отделением пользователей TEX'а (CyrTUG) под руководством Александра Бердникова. Верхняя половина этих кодировок в значительной степени повторяет кодировку T1, поэтому они вполне пригодны для печати, например, английских текстов. Александр Ходулев и Ольга Лапко разработали шрифты для каждой из кодировок T2A, T2B, T2C. Буквы русского алфавита есть в любой из этих кодировок. Табл. C.5 представляет кодировку T2A, которая для русского языка используется по умолчанию.

Список национальных языков, поддерживаемых каждой из кодировок T2A, T2B и T2C, приведен ниже.

T2A: абазинский, аварский, агульский, адыгейский, азербайджанский, алтайский, балкарский, башкирский, белорусский, болгарский, бурятский, гагаузский, даргинский, дунганский, ингушский, кабардино-черкесский, казахский, калмыцкий, каракалпакский, карачаевский, карельский, киргизский, коми, крымско-татарский, кумыкский, лакский, лезгинский, македонский, марийский, молдавский, монгольский, мордовский, ногайский, осетинский, русский, рутульский, сербский, табасаранский, таджикский, татарский, тати, телеутский, тофаларский, тувинский, туркменский, удмуртский, узбекский, украинский, ханты, цыганский, чеченский, чувашский.

T2B: абазинский, аварский, агульский, адыгейский, алеутский, алтайский, балкарский, белорусский, болгарский, бурятский, гагаузский, даргинский, долганский, дунганский, ингушский, ительменский, кабардино-черкесский, калмыцкий, каракалпакский, карачаевский,

[5] Неформальное название — Cork encoding.

карельский, кетский, киргизский, коми, корякский, крымско-татарский, кумыкский, курдский, лакский, лезгинский, мансийский, марийский, молдавский, монгольский, мордовский, нанайский, нганасанский, негидальский, ненецкий, ногайский, русский, рутульский, селькупский, табасаранский, таджикский, татарский, тати, телеутский, тофаларский, тувинский, туркменский, уйгурский, ульчи, хакасский, ханты, цыганский, чеченский, чукотский, шорский, эвенкийский, эскимосский, юкагирский, якутский.

T2C: абхазский, болгарский, гагаузский, карельский, коми, калмыкский, крымско-татарский, манси, молдавский, мордовский, нанайский, уйгурский, ногайский, русский, саамский, старо-болгарский, старо-русский, крымско-татарский, тати, телеутский, ханты, эвенкский.

Другие кодировки

Кодировки OML, OMS и OMX используются в математических формулах. Кодировка OML содержит в основном буквы из математического курсива и некоторое количество математических символов, большая часть которых сосредоточена в шрифтах с кодировкой OMS. Шрифты с кодировкой OMX служат для построения математических символов переменного размера. Конкретное распределение символов между шрифтами можно установить при помощи таблиц C.6–C.8 на стр. 453–454.

Кодировка U применяется в том случае, когда порядок символов не совпадает ни с одной известной кодировкой. Например, кодировку U имеют Эйлеровы шрифты, упоминавшиеся в главе 8[6]. Шрифты с кодировкой U представлены таблицами C.9 и C.10.

Кодировки, начинающиеся с буквы L, называются локальными. Локальные кодировки могут вводить национальные группы пользователей TEX'а. Например, предшественницей нынешних T2A, T2B, T2C была кодировка LCY.

16.3.2. Стандартные гарнитуры

Существует слишком много различных гарнитур, чтобы можно было перечислить их здесь полностью. В табл. 16.2 перечислены наиболее распространённые.

Первая буква в аббревиатуре гарнитуры обозначает производителя шрифта. Так, буква p закреплена за компанией Adobe; буквы m, t и j обозначают, соответственно, Monotype, ParaType и Microsoft. Исключение сделано для шрифтов Computer Modern. Для них зарезервировано сразу две буквы cm.

```
{\fontfamily{cmdh}\selectfont
Это гарнитура CM Dunhill.}
```

Это гарнитура CM Dunhill.

```
{\fontfamily{cmfib}\selectfont
Это гарнитура CM Fibonacci.}
```

Это гарнитура CM Fibonacci.

Некоторые гарнитуры могут существовать только в ограниченном наборе кодировок. Например, пакет bookman из коллекции PSNFSS подключает шрифты Bookman (pbk), AvantGarde (pag) и Courier (pcr) фирмы Adobe в кодировках OT1, T1 и TS1, но не T2A, поскольку в этих шрифтах нет русских букв.

[6] К счастью, кодировка всех Эйлеровых шрифтов одинакова, иначе при смене, например, насыщенности шрифта одни символы превращались бы в другие.

Таблица 16.2

Некоторые гарнитуры

family	Гарнитура	Примечание
`cmr`	CM Roman	с засечками
`cmss`	CM Sans	без засечек
`cmtt`	CM Typewriter	машинописный
`cmdh`	CM Dunhill	Данхил
`cmfib`	CM Fibonacci	Фибоначчи
`cmfr`	CM Funny	Забавный
`cmm`	CM Math Italic	математический курсив
`cmsy`	CM Math Symbols	математические символы
`cmex`	CM Math Extensions	математические знаки

Таблица 16.3

Стандартные начертания

shape	Начертание	Примечание
`n`	normal	прямой шрифт
`it`	italic	курсивный шрифт
`sl`	slanted (oblique)	наклонный шрифт
`sc`	caps and small caps	капитель
`ui`	upright italic	прямой курсивный шрифт

Таблица 16.4

Стандартные насыщенности

series	Насыщенность	Примечание
`l`	Light	светлый шрифт
`m`	Medium	нормальный шрифт
`b`	Bold	полужирный шрифт
`bx`	Bold extended	широкий полужирный шрифт
`sb`	Semi-bold	полужирный шрифт
`c`	Condensed	узкий шрифт

16.3.3. Стандартные начертания

Типичные значения параметра начертания *shape* перечислены в табл. 16.3.

```
{\fontshape{sl}\selectfont
Это начертание slanted.}
```

Это начертание slanted.

```
{\fontshape{sc}\selectfont
Это начертание Caps and Small Caps.}
```

ЭТО НАЧЕРТАНИЕ CAPS AND SMALL CAPS.

Большинство гарнитур имеют, как минимум, прямое и курсивное (или наклонное) начертания. Однако рукописные шрифты часто имеют только курсивное начертание.

16.3.4. Стандартные насыщенности

Стандартные значения параметра *series*, который задаёт насыщенность и пропорциональность шрифта, перечислены в таблице 16.3.

```
Сравните {\fontseries{b}\selectfont Bold} и
{\fontseries{bx}\selectfont Bold extended}.
```

Сравните **Bold** и **Bold extended**.

16.3.5. Стандартные размеры

Размер шрифта *size* и межстрочное расстояние *baselineskip* в декларации `\fontsize` указываются в любых единицах длины. Например, можно написать `14.4pt` или `4mm`.

```
{\fontsize{14.4pt}{4mm}\selectfont
Ничего себе размерчик!}
```

Ничего себе размерчик!

Трудно представить ситуацию, когда при подготовке печатного документа невозможно избежать применения `\fontsize`. Вместо явного задания размера шрифта посредством `\fontsize`, чреватого порчей тщательно спланированного стиля печатного документа, рекомендуется использовать декларации из ряда `\tiny`, `\scriptsize`, ... `\Huge` (раздел 1.11 и таблица 16.6 на стр. 379). Они позволяют выбирать шрифты заданных размеров.

В профессиональных издательских системах обычно доступны кегли 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 24, 30 и 36. Отсутствующие размеры получают линейным масштабированием. Декларации переключения размера шрифта (от `\tiny` до `\Huge`) обычно оперируют с размерами 5, 6, 7, 8, 9, 10, 10.95, 12, 14.4, 17.28, 20.74, 24.88 пунктов, которые приблизительно составляют геометрическую прогрессию с множителем 1,2. Шрифты спроектированы для размеров 5, 6, 7, 8, 9, 10, 12 и 17 пунктов, а отсутствующие размеры получают линейным масштабированием ближайшего из приведённых значений. Другие шрифты могут иметь иной набор проектных размеров.

Линейное масштабирование шрифтов немного ухудшает качество воспроизведения текста, поскольку трекинг не должен изменяться пропорционально высоте букв, а кроме того, шрифты мелких кеглей относительно шире, чем шрифты больших кеглей. Текст, набранный кеглем 5, после увеличения в 2 раза будет заметно шире, чем тот же текст, набранный кеглем 10.

Точное соответствие деклараций переключения параметрам шрифта определяется выбранным классом печатного документа. Все стандартные классы вводят специальные пользовательские команды, которые мы рассмотрим в следующем разделе.

16.4. Пользовательские команды

Типичные варианты переключения гарнитуры, насыщенности и начертания шрифта обслуживаются более простыми командами, нежели `\fontcmd`. Эти команды так называемого пользовательского уровня и соответствующие им атрибуты перечислены в табл. 16.5. Например, вместо `\usefont{OT1}{cmr}{m}{it}` обычно достаточно напи-

Таблица 16.5

Пользовательские команды переключения шрифта и соответствующие им атрибуты

Команда	Декларация	Атрибут	Значение по умолчанию	
\textrm{*text*}	\rmfamily	*family*	\rmdefault	(cmr)
\textsf{*text*}	\sffamily	*family*	\sfdefault	(cmss)
\texttt{*text*}	\ttfamily	*family*	\ttdefault	(cmtt)
\textmd{*text*}	\mdseries	*series*	\mddefault	(m)
\textbf{*text*}	\bfseries	*series*	\bfdefault	(bx)
\textup{*text*}	\upshape	*shape*	\updefault	(n)
\textit{*text*}	\itshape	*shape*	\itdefault	(it)
\textsl{*text*}	\slshape	*shape*	\sldefault	(sl)
\textsc{*text*}	\scshape	*shape*	\scdefault	(sc)
\emph{*text*}	\em	*shape*		
\textnormal{*text*}	\normalfont	ENC	\encodingdefault	(OT1)
		family	\familydefault	(\rmdefault)
		series	\seriesdefault	(\mddefault)
		shape	\shapedefault	(\updefault)

сать \itshape. Однако теперь можем объяснить, что скрывается за ширмой простых команд типа \textit.

Соответствие пользовательских команд и деклараций достаточно очевидно. Команда \textit{...} эквивалентна конструкции {\itshape ...\/}, причём \/ вставляет корректирующий пробел при переходе от наклонного или курсивного шрифта к прямому (раздел 4.3).

Декларация \itshape действует менее прямолинейно, а именно: она переключает атрибут начертания *shape* в состояние, которое хранится в команде \itdefault. Иными словами, она равнозначна комбинации

```
\fontshape{\itdefault}\selectfont
```

По умолчанию \itdefault определена как it, но достаточно изменить её на ui, как наклонный курсив превратится в прямой (если тот имеется в используемой гарнитуре).

```
Был \textit{наклонный} курсив,
\renewcommand{\itdefault}{ui}
стал \textit{прямой}.
```

Был *наклонный* курсив, стал прямой.

Другие пользовательские команды переключения шрифтов также реализованы через промежуточные команды *cmd*default, как показано в табл. 16.5. В последней колонке этой таблицы в скобках указаны значения атрибутов, используемые по умолчанию в стандартных классах печатных документов. В частности, команды \textrm, \textsf и \texttt печатают текст шрифтом заданной гарнитуры. Им соответствуют декларации \rmfamily, \sffamily и \ttfamily. Они инструктируют компилятор, что от текущей позиции до конца текущей группы фигурных или командных скобок

нужно использовать шрифт выбранной заранее гарнитуры, аббревиатура которой хранится соответственно в декларациях `\rmfamily`, `\sffamily` и `\ttfamily`. Команды рекомендуется использовать для выделения отдельного слова или короткой фразы, а декларации лучше использовать при определении новых команд или процедур. Для выделения длинных абзацев удобнее использовать процедурную форму деклараций, причём имя процедуры получается удалением обратного слеша из имени декларации. Например, чтобы выделить несколько абзацев шрифтом без засечек, их нужно поместить между `\begin{sffamily}` и `\end{sffamily}`.

Чтобы напечатать документ шрифтами Adobe Bookman, AvantGarde и Courier, в преамбулу входного файла достаточно поместить декларации

```
\renewcommand{\rmdefault}{pbk}
\renewcommand{\sfdefault}{pag}
\renewcommand{\ttdefault}{pcr}
```

Другой способ достичь той же цели — загрузить пакет bookman из коллекции PSNFSS; он делает буквально те же манипуляции с наименованиями гарнитур. При всех возможных переопределениях правил загрузки шрифтов обычно стремятся к тому, чтобы команда `\rmdefault` указывала на гарнитуру с засечками, `\sfdefault` — без засечек, а `\ttdefault` соответствовала машинописному шрифту. В данной главе значение `\rmdefault` изменено загруженным пакетом TimesC:

```
\verb|\rmdefault|=\rmdefault\\
\verb|\sfdefault|=\sfdefault\\
\verb|\ttdefault|=\ttdefault
```

\rmdefault=mnt
\sfdefault=cmss
\ttdefault=cmtt

Большинство шрифтов ограничено двумя вариантами насыщенности: нормальной и полужирной. Следуя этой традиции, LaTeX на уровне пользовательских команд также ограничивается этими двумя вариантами: `\textmd` и `\textbf`. Двум командам соответствуют две декларации: `\mdseries` и `\bfseries`. Используемые ими атрибуты хранятся в командах `\mddefault` и `\bfdefault`. Первая содержит атрибут нормальной насыщенности, вторая — полужирной. Как указано в табл. 16.5, в стандартных классах `\bfdefault` имеет значение `bx`. Некоторые шрифты PostScript имеются только в варианте `b`. Поэтому пакеты, спроектированные для работы с этими шрифтами, присваивают параметру `\bfdefault` значение `b`. Впрочем, если компилятор не находит шрифт серии `b`, он автоматически подменяет его шрифтом серии `bx`, и наоборот. Если же какая-либо гарнитура имеется во множестве вариантов насыщенности, то «добраться» до них в одном документе невозможно без помощи команд низшего уровня, как показано в примере на странице 375.

Четыре команды: `\textup`, `\textit`, `\textsl` и `\textsc` — изменяют форму шрифта, проводя выбор из четырёх наиболее распространённых начертаний: прямого, курсивного, наклонного и капители. Им соответствуют четыре декларации: `\upshape`, `\itshape`, `\slshape` и `\scshape`. Ещё четыре команды: `\updefault`, `\itdefault`, `\sldefault` и `\scdefault` — хранят атрибуты начертаний.

Команда `\emph` и соответствующая ей декларация `\em` используются для выделения текста. Они изменяют начертание шрифта так, чтобы он отличался от окружаю-

щего текста. Если до \emph использовался прямой шрифт (\upshape или \scshape), то выделенный текст печатается курсивом (\itshape). В остальных случаях \emph включает прямой шрифт (\upshape). Команды \emph могут быть вложены друг в друга очевидным способом:

`Выделенный \emph{текст \emph{внутри} курсива} печатается прямым шрифтом.`	Выделенный *текст* внутри *курсива* печатается прямым шрифтом.

Однако не следует злоупотреблять выделением фрагментов текста. Это рассеивает внимание читателя, вызывая эффект, обратный желаемому.

Гарнитуру, насыщенность и начертание шрифта можно изменять независимо, но из двух подряд деклараций переключения гарнитуры (или насыщенности, или начертания), разумеется, действует последняя.

Наконец, команда \textnormal и соответствующая ей декларация \normalfont выбирают *главный шрифт* документа, которым печатается его бóльшая часть. Компилятор автоматически выбирает главный шрифт вне области действия любых деклараций переключения шрифтов. Поэтому команды \textnormal и \normalfont чаще всего используются при определении новых команд или процедур, когда важно, чтобы текст был напечатан одним и тем же шрифтом независимо от окружения. Например, аргументы команд в нашей книге печатает команда \m, которая определена следующим образом:

```
\newcommand{\m}[1]{{\normalfont\fontshape{ui}\selectfont #1}}
```

Наличие \normalfont здесь гарантирует, что никакие ухищрения не могут изменить шрифт, который выбирает команда \m для печати своего аргумента:

`\m{family}, \texttt{\m{family}}, \textbf{\m{family}}`	*family*, *family*, *family*

Гарантированный результат достигается тем, что \normal переключает все атрибуты шрифта, кроме его размера. Иными словами, её действие эквивалентно команде

```
\usefont{\encodingdefault}{\familydefault}{\seriesdefault}{\shapedefault}
```

где \encodingdefault, \familydefault, \seriesdefault, \shapedefault — суть команды, которые соответственно указывают на кодировку, гарнитуру, насыщенность и начертание шрифта, используемого по умолчанию. Формат LaTeX содержит следующие определения этих команд:

```
\newcommand{\encodingdefault}{OT1}
\newcommand{\familydefault}{\rmdefault}
\newcommand{\seriesdefault}{\mddefault}
\newcommand{\shapedefault}{\updefault}
```

Но сейчас мы имеем следующую картину:

`\verb|\encodingdefault|=\encodingdefault\\`	\encodingdefault=T2A
`\verb|\familydefault|=\familydefault\\`	\familydefault=mnt
`\verb|\seriesdefault|=\seriesdefault\\`	\seriesdefault=m
`\verb|\shapedefault|=\shapedefault`	\shapedefault=n

Таблица 16.6

Соответствие деклараций переключения размера шрифта и параметров size/baselineskip

Декларация	Опция 10pt	Опция 11pt	Опция 12pt
\tiny	5pt / 6pt	6pt / 7pt	6pt / 7pt
\scriptsize	7pt / 8pt	8pt / 9.5pt	8pt / 9.5pt
\footnotesize	8pt / 9.5pt	9pt / 11pt	10pt / 12pt
\small	9pt / 11pt	10pt / 12pt	11pt / 13.6pt
\normalsize	10pt / 12pt	11pt / 13.6pt	12pt / 14.5pt
\large	12pt / 14pt	12pt / 14pt	14pt / 18pt
\Large	14pt / 18pt	14pt / 18pt	17pt / 22pt
\LARGE	17pt / 22pt	17pt / 22pt	20pt / 25pt
\huge	20pt / 25pt	20pt / 25pt	25pt / 30pt
\Huge	25pt / 30pt	25pt / 30pt	25pt / 30pt

Кодировку по умолчанию изменила загрузка пакета babel с опцией `russian`. Гарнитуру заменил уже упоминавшийся пакет TimesC, поскольку он содержит строчку

```
\renewcommand{\rmdefault}{mnt}
```

Таким образом, чтобы изменить гарнитуру главного шрифта печатного документа, достаточно переопределить `\rmdefault`. Предлагаем Читателю самостоятельно разобрать маленький фокус, который произойдёт, если, не затрагивая `\rmdefault`, записать

```
\renewcommand{\familydefault}{mnt}
```

Соответствие между размером шрифта и значением аргументов size и baselineskip декларации `\fontsize` для стандартных классов печатного документа указано в табл. 16.6. В зависимости от выбранной опции класса `10pt`, `11pt`, `12pt` возможны три варианта соответствия, приведённые в колонках таблицы.

Наконец отметим, что на пользовательском уровне нет специальных команд для выбора кодировки шрифта, поскольку в обычном печатном документе нет необходимости производить динамическое переключение кодировки. Для печати всего документа в целом шрифтами с кодировкой, отличной от используемой по умолчанию, следует загрузить пакет fontenc, описанный в разделе 16.5.

16.5. Замена кодировки

LaTeX допускает использование нескольких кодировок в одном документе. Множественность внутренней кодировки — скорее норма, чем исключение. Например, в математических формулах используются обычно все имеющиеся математические кодировки, хотя процесс переключения кодировок там скрыт от пользователя. Пара текстовых кодировок в одном документе — также не столь уж большая редкость, но и здесь

пользователь может не догадываться, что некоторые символы взяты из разных кодировок. Совершенной экзотикой является применение нескольких внешних кодировок в одном исходном файле, однако и такое возможно. Явное переключение кодировок требуется в документах, наподобие нашей книги, где поневоле приходится «выжимать педаль газа до упора». В этом разделе мы расскажем, как это сделать.

16.5.1. Внутренняя кодировка

Команда

```
\symbol{code}
```

печатает символ с кодом code, соответствующим порядковому номеру позиции символа в текущей внутренней кодировке шрифта. Код может быть выражен десятичным числом от 0 до 255, его восьмеричным или шестнадцатеричным эквивалентом, которому должны предшествовать, соответственно, апострофы ' и ".

```
\symbol{130}  \symbol{146}  \symbol{150}  \\
\symbol{'202} \symbol{'222} \symbol{'226} \\
\symbol{"82}  \symbol{"92}  \symbol{"96}
```

Ђ Ў Џ
Ђ Ў Џ
Ђ Ў Џ

В документах на английском языке (как и на многих других европейских языках) по умолчанию используется кодировка OT1, хотя она считается устаревшей. Ей на замену предназначена кодировка T1. Чтобы она стала основной кодировкой печатного документа, достаточно загрузить пакет fontenc с опцией T1:

```
\usepackage[T1]{fontenc}
```

Пакет fontenc уникален в том смысле, что его можно загружать несколько раз и каждый раз с новым значением необязательного аргумента, тогда как повторная загрузка других пакетов с различающимся набором опций считается ошибкой. Если в необязательном аргументе \usepackage перечислены две, три или большее число кодировок, основной кодировкой будет объявлена последняя, но будут сделаны необходимые приготовления для применения всех кодировок. Например,

```
\usepackage[T2A,T1]{fontenc}
```

подготавливает использование в одном входном файле кодировок T2A и T1. Подготовка заключается в том, что происходит определение команд, специфичных для кодировок T2A и T1 (см. главу 4). Пакет babel в зависимости от выбранного языка может самостоятельно выбрать подходящую кодировку. Например, для русского языка он выбирает кодировку T2A. Чтобы выбрать иную кодировку, её нужно указать *до* загрузки пакета babel:

```
\usepackage[T2B,T1]{fontenc}
\usepackage[english,russian]{babel}
```

При этом порядок перечисления кодировок при загрузке пакета fontenc уже не играет роли, поскольку пакет babel разберется, какую кодировку выбрать для каждого языка.

Выбор кодировок в преамбуле входного файла рекомендуется делать до загрузки шрифтовых пакетов. В некоторых случаях это позволяет избежать загрузки шрифтов, которые реально не используются.

При переключении языка внутри документа не нужно явно переключать и кодировку, поскольку команда `\selectlanguage` (или её аналоги, рассмотренные в разделе 3.6) делает это сама:

```
\verb|\encodingdefault|\encodingdefault\\
\selectlanguage{english}
\verb|\encodingdefault|=\encodingdefault
```

\encodingdefault=T2B
\encodingdefault=T1

Явное переключение кодировки можно использовать, чтобы добраться до символа, отсутствующего в текущей кодировке:

```
Команда \verb|\dj| ({\fontencoding{T1}\selectfont
\dj}) определена в кодировке \texttt{T1}.
```

Команда `\dj` (đ) определена в кодировке T1.

Переключение кодировки есть более экономная операция, чем переключение языка.

16.5.2. Внешняя кодировка

Клавиатуры компьютеров, адаптированные к национальным языкам, позволяют набирать все буквы национального алфавита, например греческие буквы или буквы с диакритическими знаками. Пакет inputenc позволяет пользователю конкретизировать внешнюю кодировку, используемую операционной системой компьютера, и переложить на компилятор работу по трансляции символов, доступных с клавиатуры, во внутренние команды LaTeX'а. Такая трансляция неизбежна, поскольку LaTeX способен работать со множеством языков, используя лишь ограниченный набор внутренних кодировок. Например, жители Восточной Европы, работающие с кодовой страницей 852, могут информировать об этом LaTeX, загрузив пакет inputenc с опцией cp852:

```
\usepackage[cp852]{inputenc}
```

Коллекция пакетов cyrillic, которая должна быть обязательной частью любой реализации системы LaTeX, содержит все существующие или существовавшие кодировки с наличием букв русского алфавита. Полный список возможных опций пакета inputenc приведен в табл. 16.7. Например, чтобы откомпилировать исходный текст, подготовленный на русском языке в среде Windows, достаточно включить в преамбулу входного файла команду

```
\usepackage[cp1251]{inputenc}
```

Внутренняя кодировка T2A столь близка к кодовой странице cp1251, что исходные тексты, подготовленные в русской версии операционной системы Windows, можно сравнительно успешно компилировать без загрузки пакета inputenc[7].

[7] Но при этом будет потеряна буква «ё».

Таблица 16.7

Внешние кодировки

Опция	Кодировка
`applemac`	Macintosh
`ascii`	ASCII (для кодов 32–127)
`decmulti`	DEC Multinational Character Set
`cp437`	IBM 437 (MS DOS, Европа)
`cp437de`	IBM 437 (MS DOS, Германия)
`cp850`	IBM 850 (MS DOS, Европа)
`cp852`	IBM 852 (MS DOS, Центральная Европа)
`cp855`	IBM 855 (MS DOS, Европа) 1)
`cp865`	IBM 865 (MS DOS, Норвегия)
`cp866`	IBM 866 (MS DOS, Россия) 1) 2)
`cp1250`	Windows (Центральная Европа)
`cp1251`	Cyrillic Windows (Россия, Украина, Белорусия) 1)
`cp1252`	Windows (Европа, Германия)
`iso88595`	ISO-8859-5 (UNIX, Россия, Украина, Белорусия) 1)
`isoir111`	ISO-IR-111 ECMA Cyrillic (UNIX, Россия, Украина, Белорусия) 1)
`koi8-r`	Russian Net Character Set (UNIX, Россия, Украина, Белорусия) 1)
`koi8-ru`	Cyrillic KOI8 (UNIX, Россия, Украина, Белорусия) 1)
`koi8-u`	Extended KOI8-R and ISO-IR-111 (UNIX, Россия, Украина, Белорусия) 1)
`latin1`	ISO 8859-1 (Западная Европа)
`latin2`	ISO 8859-2 (Восточная Европа)
`latin3`	ISO 8859-3 (другие европейские языки)
`latin4`	ISO 8859-4 (Северная Европа, Балтия)
`latin5`	ISO 8859-9 (Турция)
`latin9`	Latin-9 (Западная Европа) 3)
`maccyr`	Cyrillic Mac (Россия) 1)
`macukr`	Cyrillic Mac (Украина) 1)
`ncc`	NCC 1) 4)
`next`	Next

1) Распространяется в составе коллекции пакетов cyrillic.

2) Имеются также другие варианты данной кодировки: `cp866av`, `cp866mav`, `cp866nav`, `cp866tat`.

3) На замену кодировки `latin1`. Содержит знак евро.

4) Имеются также другие кодировки, о происхождении которых сейчас мало кто помнит: `ctt`, `dbk`, `mik`, `mls`, `mnk`, `mos`.

Объявить об изменении внешней кодировки посреди входного файла можно при помощи декларации

```
\inputencoding{ext-encoding}
```
(inputenc)

В её аргументе *ext-encoding* должна быть одна из перечисленных в табл. 16.7 опций пакета inputenc.

Существует альтернативный метод преобразования кодировки исходного текста во внутреннюю кодировку LaTeX'а. Это механизм TCX (TeX character translation). Он использует файлы (обычно имеющие расширение `tcx`), которые устанавливают однозначное соответствие кодов каждого символа во внешней и внутренней кодировках. Имя `tcx`-файла передаётся компилятору через значение ключа командной строки. Механизм TCX поддерживают не все компиляторы. Он менее гибок, нежели загрузка пакета inputenc, так как не позволяет в одном документе использовать несколько кодировок (внутренних или внешних) одновременно. Более существенным недостатком метода TCX считается то, что автор не обязан декларировать кодировку исходного текста во входном файле, что может затруднить компиляцию документа на компьютере, работающем под управлением иной операционной системы. Однако у метода TCX есть свойство, которое в определённых ситуациях может предстать достоинством. В частности, использование перекодировки TCX позволяет избавиться от надоедливых команд `\cyr`*cmd*, подменяющих буквы русского алфавита в служебных файлах, которые генерирует компилятор. Это может пригодиться, например, при составлении алфавитного указателя (глава 14).

16.6. Шрифты для формул

Переключение шрифтов в математической моде имеет кардинальные отличия от того, как это делается в обычном тексте. Выбор некоторых математических шрифтов осуществляется явно при помощи команд вида `\mathsf{...}`, `\mathbf{...}` и т. д. (раздел 6.6) с одним аргументом. Такие команды называются *математическими алфавитами*. Другие шрифты неявно выбираются компилятором при обработке команд, печатающих математические символы, такие как `\oplus` ($\oplus$) или `+`. Такие шрифты называются *символьными*.

16.6.1. Математические алфавиты

Стандартные классы содержат определения следующих математических алфавитов:

```
\mathrm{math}      \mathbf{math}      \mathsf{math}
\mathit{math}      \mathtt{math}      \mathcal{math}
\mathnormal{math}
```

Мы не будем здесь повторять примеры их использования, отсылая Читателя к разделу 6.6. Напомним только, что математические алфавиты воздействуют только на буквы и цифры; все другие символы в их аргументах остаются без изменений.

Таблица 16.8

Символьные шрифты

Название	Кодировка	Пример	Примечание
operators	OT1	[+]	прямые символы из \mathrm
letters	OML	$<<\star>>$	курсивные символы из \mathnormal
symbols	OMS	$\leq * \geq$	математические символы
largesymbols	OMX	$\sum\prod\int$	большие символы

Набор алфавитов дополняется различными пакетами. Например, пакет amsfonts вводит алфавиты \mathfrak и \mathbb для печати соответственно готических и контурных букв (раздел 8.3).

Перед использованием русских букв в математических формулах необходимо загрузить пакет mathtext из коллекции T2, причём сделать это нужно до загрузки пакетов, изменяющих внутреннюю кодировку, таких как fontenc или babel. Выполнив все эти рекомендации, мы можем предъявить следующий пример:

```
Разве $dЮ = \mathit{dЮ}$?
```

Разве dЮ = *dЮ*?

Здесь в левой части уравнения использован символьный шрифт, а в правой — математический алфавит. Пакет mathtext устанавливает, что русские буквы в символьных шрифтах имеют прямое начертание в отличие от латинских, которые всегда печатаются курсивом.

Шрифтовые пакеты могут вводить дополнительные русифицированные математические алфавиты или переопределять существующие. Например, пакет mathmnt из коллекции FontsC подменяет шрифты в стандартных алфавитах, так что русские буквы там появляются даже без загрузки пакета mathmnt (раздел 16.10).

16.6.2. Символьные шрифты

Непосредственно в математической формуле[8] LaTeX использует так называемые *символьные* шрифты с кодировками OML, OMS и OMX. Часть символов, которые имеют прямое начертание, заимствуется из текстовых шрифтов, так что всего имеется 4 типа символьных шрифтов (табл. 16.8). Символьные шрифты имеют те же пять атрибутов (кодировка ENC, гарнитура *family*, насыщенность *series*, начертание *shape* и размер *size*), что и текстовые. Однако для математических формул нет команд, которые бы переключали только один атрибут, поскольку формула — это единое целое, не допускающее вольностей. По этой причине комбинация математических алфавитов, как отмечалось в разделе 6.6, не даёт новый шрифт.

Соответствие алфавита шрифту с заданным перечнем атрибутов контролирует *математическая версия*. Версию изменяет декларация

[8] Вне аргументов команд математических алфавитов, а также внутри этих аргументов для символов, не являющихся буквами или цифрами.

Таблица 16.9

Соответствие атрибутов внешнему шрифту

Атрибуты шрифта	Внешний шрифт и его название	
`OT1 cmr m n 10pt`	`cmr10 at 10pt`	CM Roman
`OT1 cmss m sl 12pt`	`cmssi12 at 12pt`	CM Sans Slanted
`OML cmm m it 10pt`	`cmmi10 at 10pt`	CM Math Italic
`T1 ptm b it 18pt`	`ptmbi8t at 18pt`	Adobe Times Bold Italic
`T2A cmr m n 10pt`	`lhr10 at 10pt`	LH Roman
`T2A cmss m sl 12pt`	`lhssi12 at 12pt`	LH Sans Slanted

`\mathversion{`*version-name*`}`

где *version-name* — название версии. Её можно менять только вне математической формулы. Стандартные классы предопределяют две версии `normal` и `bold`; первая из них используется по умолчанию.

Изменение версии обычно влияет не только на буквы и цифры, но и на другие символы, поскольку сопровождается заменой реального, так называемого *внешнего* шрифта.

16.7. Внешний шрифт

Результат переключения шрифтов был бы легко предсказуем, если бы все гарнитуры имелись в одинаковом наборе начертаний и насыщенности. Но это невозможно в принципе. Например, рукописные шрифты имеются обычно только в варианте светлого курсива.

Что же произойдёт, если шрифт с запрошенным набором атрибутов отсутствует? Система NFSS предпримет попытку разрешить возникшую проблему, руководствуясь информацией, записанной в файлах определения шрифтов (font definition) с расширением `fd`. Название файла образуется объединением аббревиатур кодировки ENC и гарнитуры *family*. Например, если заказана гарнитура `cmr` (Computer Modern Roman) в кодировке `T2A`, компилятор попытается найти файл `t2acmr.fd`. Синтаксис деклараций, которые могут использоваться в файлах определения шрифтов, несложен, но его описание выходит за рамки нашей книги (см. [12, 20]). Несколько примеров соответствия атрибутов внешнему шрифту приведены в табл. 16.9. Если комбинация кодировки ENC, семейства *family*, насыщенности *series* и начертания *shape* не определена, то NFSS меняет значение атрибута начертания на принятое по умолчанию. Если получившаяся комбинация по-прежнему неизвестна, значение по умолчанию присваивается атрибуту насыщенности *series*. В качестве последнего противоядия предпринимается попытка изменить атрибут *family*. Замена кодировки была бы чревата искажением смысла текста из-за появления «не тех» букв. Таков типичный сценарий действий NFSS, но и он может быть изменён.

Например, если запросить рукописный шрифт в прямом начертании нормальной насыщенности (жирного рукописного шрифта всё равно нет), то NFSS, действуя по стандартной схеме, подменит его шрифтом другого семейства. Действительно, поскольку прямое начертание — это начертание по умолчанию, равно как и нормальная насыщенность есть насыщенность по умолчанию, NFSS сразу переключит атрибут семейства. Разработчики шрифтового пакета могут задать иные правила подмены (всё в том же файле определения шрифтов или в самом пакете) с тем, чтобы предотвратить подмену гарнитуры:

```
{\usefont{T1}{pzc}{m}{it} ZapfChancery}\\
{\usefont{T1}{pzc}{m}{n} ZapfChancery}
```

ZapfChancery
ZapfChancery

Большинство операций со шрифтами, производимые в процессе компиляции документа, фиксируются в файле протокола (с расширением `log`). Расшифровка этих сообщений приведена в приложении В. Подробность сообщений можно регулировать, если загрузить пакет tracefnt с различными опциями. Вот эти опции, перечисленные в порядке возрастания подробности сообщений: `errorshow`, `warningshow`, `infoshow`, `debugshow`, `loading`, `pausing`.

16.8. Пакеты PSNFSS

Начало коллекции PSNFSS было положено Себастьяном Ратцем (Rahtz, Sebastian). Наряду с поддержкой упоминавшегося в начале главы набора 35 PostScript-шрифтов фирмы Adobe коллекция PSNFSS содержала пакеты для подключения множества других PostScript-шрифтов, которые большей частью необходимо приобретать за отдельную плату. В настоящее время поддержка коммерческих шрифтов выделена в самостоятельную коллекцию PSNFSSx. Среди пакетов PSNFSSx следует выделить пакеты hvmaths, lucidabr, ly1, mathtime, mtpro, tmmaths, которые предназначены для замены математических шрифтов Computer Modern на современные шрифты Lucida Bright и MathTime, продаваемые компаниями Y&Y[9] и MicroPress[10]. Мы сосредоточим внимание Читателя на некоммерческих шрифтах, поддерживаемых пакетами PSNFSS, тем более что здесь также есть замечательные шрифты семейств Palatino и Times для математических формул. В настоящее время пакеты PSNFSS сопровождает Вальтер Шмидт (Schmidt, Walter).

Установка всего необходимого для работы с пакетами PSNFSS, включая шрифты, может потребовать изучения сопроводительной документации. Мы можем только засвидетельствовать, что в полном комплекте библиотеки MiKTeX оказалось всё, что нужно, но в «разобранном виде», а именно: собственно коллекция PSNFSS, шрифты Courier, Utopia компании Adobe, Charter компании Bitstream, PazoMath Диего Пуги (Puga, Diego) — находились в разных архивах. Что касается 35 шрифтов Adobe, то их бесплатный клон входит в состав программы Ghostscript, которая является почти обязательной составной частью любой современной реализации системы LaTeX. Во вся-

[9] Адрес в интернете: `http://www.yandy.com`.
[10] Адрес в интернете: `http://www.micropress-inc.com`.

Таблица 16.10

Гарнитуры, используемые пакетами PSNFSS версии 9.0.
Пропуск означает, что гарнитура не изменяется пакетом

Пакет	\rmdefault	\sfdefault	\ttdefault	Формулы
	cmr	cmss	cmtt	≈ CM Roman
avant		pag		
bookman	pbk	pag	pcr	
chancery	pzc			
charter	bch			
courier			pcr	
helvet		phv		
mathpazo	ppl			≈ Palatino
mathptmx	ptm			≈ Times
newcent	pnc	pag	pcr	
pifont				
utopia	put			

Таблица 16.11

Шрифты, поддерживаемые пакетами PSNFSS версии 9.0

ENC	family	series	shape	Гарнитура
OT1 T1 TS1	ptm	m b	n sl it sc	Times
OT1 T1 TS1	ppl	m b	n sl it sc	Palatino
OT1 T1 TS1	pnc	m b	n sl it sc	NewCenturySchlbk
OT1 T1 TS1	pbk	m b	n sl it sc	Bookman
OT1 T1 TS1	phv	m b mc bc	n sl sc	Helvetica
OT1 T1 TS1	pag	m b	n sl sc	AvantGarde
OT1 T1 TS1	pcr	m b	n sl sc	Courier
OT1 T1 TS1	pzc	m	it	*ZapfChancery*
OT1 T1 TS1	put	m b	n sl it sc	Utopia
OT1 T1 TS1	bch	m b	n sl it sc	CharterBT
U	psy	m	n	Σψμβολ (Symbol)
U	pzd	m	n	✺❁❐❆⌘✻■❈✷❁▼▲ (ZapfDingbats)

ком случае, все необходимые шрифты можно получить с серверов CTAN из каталога /fonts.

Чтобы изменить шрифт, пользователь выбирает один из основных пакетов avant, bookman, charter, chancery, helvet и т. д. Они заменяют одну или сразу несколько гарнитур, используемых по умолчанию. Напомним, что аббревиатуры гарнитур хранятся в командах \rmdefault, \sfdefault, \ttdefault (с засечками, рубленые и машинописные шрифты, соответственно). Табл. 16.10 показывает, какую замену производит каждый пакет из коллекции PSNFSS. Два из них: mathpazo и mathptmx подменяют шрифты не только в тексте, но и в математических формулах. Соответствие аббревиа-

тур названиям реальных шрифтов устанавливает табл. 16.11, причём названия набраны шрифтами соответствующей гарнитуры (если название невозможно прочитать в «своей» гарнитуре, оно продублировано в скобках). Все перечисленные в таблице гарнитуры имеются в кодировках `OT1`, `T1` и `TS1`. Исключение составляют две последние гарнитуры Symbol и ZapfDingbats, которые имеются только в кодировке `U`.

Для документов на русском языке пакеты PSNFSS не подходят, поскольку в них нет русских букв. Поэтому мы не будем более детально рассматривать эти пакеты, сделав исключение для пакетов helvet и pifont.

Гарнитура Helvetica несколько крупнее, чем другие гарнитуры при номинально равных размерах. Как следствие, смешивание Helvetica с другими гарнитурами в одной строке делает текст эстетически малопривлекательным. Проблему решает загрузка пакета helvet с опцией `[scaled=`*scale*`]`. Например,

```
\usepackage[scaled=0.92]{helvet}
```

масштабирует гарнитуру Helvetica на 92% номинального размера. Значение параметра *scale* (вместо со знаком =) можно опустить, что соответствует масштабу 95%.

Пакет pifont не производит подмену гарнитур, поскольку предназначен для иных целей. Ему посвящён следующий раздел.

16.8.1. Пакет pifont

Пакет pifont из коллекции PSNFSS вводит несколько команд и процедур, которые используют шрифты Adobe ZapfDingbats и Symbol. Начнём с команд:

```
\ding{code}
\dingfill{code}
\dingline{code}
```
(pifont)

Первая из них просто печатает символ Dingbat с кодом *code*:

```
\ding{"76} это Dingbat "76
```
❖ это Dingbat "76

Соответствие символов и кодов нетрудно установить с помощью табл. C.10 на стр. 455. Команда `\dingfill` аналогична `\dotfill` (раздел 4.3), но вместо точек печатает указанный символ, а команда `\dingline` заполняет отдельную строку заданным символом и может служить для визуального разделения печатного документа на части.

```
A\dingfill{101}Z
\dingline{36}
```
A ❋ ❋ ❋ ❋ ❋ ❋ ❋ ❋ Z

✄ ✄ ✄ ✄

Процедура

```
\begin{dinglist}{code} ... \end{dinglist}
```
(pifont)

форматирует список аналогично `itemize` (раздел 5.4), но в качестве метки каждой записи печатает символ Dingbat с кодом *code*.

```
\begin{dinglist}{"4A}
  \item 1-я запись 1-го уровня
    \begin{dinglist}{"2E}
      \item 1-я запись 2-го уровня
      \item 2-я запись 2-го уровня
    \end{dinglist}
  \item 2-я запись 1-го уровня
\end{dinglist}
```

✪ 1-я запись 1-го уровня

✎ 1-я запись 2-го уровня

✎ 2-я запись 2-го уровня

✪ 2-я запись 1-го уровня

Более сложные команды

```
\Pisymbol{family}{code}
\Pifill{family}{code}
\Piline{family}{code}
```

(pifont)

аналогичны соответственно `\ding`, `\dingfill` и `\dingline`, позволяя выбрать символ из заданной гарнитуры.

```
\Pisymbol{pzd}{"76}
```

❖

То же самое относится к процедурам

```
\begin{Pilist}{family}{code} ... \end{Pilist}
\begin{Piautolist}{family}{code} ... \end{Piautolist}
```

(pifont)

причём во второй из них код символа, используемого в качестве метки, увеличивается на 1 для каждой следующей записи того же уровня.

```
\begin{Piautolist}{pzd}{192}
  \item 1-я запись 1-го уровня \label{PI}
    \begin{Piautolist}{pzd}{"2E}
      \item 1-я запись 2-го уровня
      \item 2-я запись 2-го уровня
    \end{Piautolist}
  \item 2-я запись 1-го уровня
\end{Piautolist}
Ссылка на пункт \ref{PI}.
```

① 1-я запись 1-го уровня

✎ 1-я запись 2-го уровня

✏ 2-я запись 2-го уровня

② 2-я запись 1-го уровня

Ссылка на пункт ①.

Как видно из примера, ссылки на записи в списке формируются обычным способом при помощи команд `\label` и `\ref`.

16.9. Пакеты pxfonts и txfonts

Пакеты pxfonts и txfonts подменяют все шрифты в документе, как в тексте, так и в математических формулах, придавая документу хорошо сбалансированный вид. Пакеты и шрифты PostScript для них разработал Янг Рю (Ryu, Young). Пакет pxfonts подключает шрифты Palatino, а пакет txfonts — шрифты Times. Те и другие разработаны на основе одноимённых шрифтов фирмы Adobe. В отличие от пакетов коллекции

Таблица 16.12

Гарнитуры, используемые пакетами pxfonts и txfonts.

Пакет	\rmdefault	\sfdefault	\ttdefault	Формулы
	cmr	cmss	cmtt	≈ CM Roman
pxfonts	pxr	pxss	pxtt	≈ Palatino
txfonts	txr	txss	txtt	≈ Times

Таблица 16.13

Шрифты, поддерживаемые пакетами pxfonts и txfonts

ENC	family	series	shape	Гарнитура
OT1 T1 TS1	pxr	m bx	n sl it sc	PX Roman (Palatino)
OT1 T1 TS1	pxss	m bx	n sl sc	PX Sans Serif (Palatino)
OT1 T1 TS1	pxtt	m bx	n sl sc	PX Typewriter (Palatino)
OT1 T1 TS1	txr	m bx	n sl it sc	TX Roman (Times)
OT1 T1 TS1	txss	m bx	n sl sc	TX Sans Serif (Times)
OT1 T1 TS1	txtt	m bx	n sl sc	TX Typewriter (Times)

Таблица 16.14

Символы бинарных операций (пакеты pxfonts и txfonts)

$\medcirc$	\medcirc	$\medbullet$	\medbullet	$\invamp$	\invamp
$\circledwedge$	\circledwedge	$\circledvee$	\circledvee	$\circledbar$	\circledbar
$\circledbslash$	\circledbslash	$\nplus$	\nplus	$\boxast$	\boxast
$\boxbslash$	\boxbslash	$\boxbar$	\boxbar	$\boxslash$	\boxslash
$\Wr$	\Wr	$\sqcupplus$	\sqcupplus	$\sqcapplus$	\sqcapplus
$\rhd$	\rhd	$\lhd$	\lhd	$\unrhd$	\unrhd
$\unlhd$	\unlhd				

Таблица 16.15

Символы сравнения (пакеты pxfonts и txfonts)

$\mappedfrom$	\mappedfrom	$\longmappedfrom$	\longmappedfrom
$\Mapsto$	\Mapsto	$\Longmapsto$	\Longmapsto
$\Mappedfrom$	\Mappedfrom	$\Longmappedfrom$	\Longmappedfrom
$\mmapsto$	\mmapsto	$\longmmapsto$	\longmmapsto
$\mmappedfrom$	\mmappedfrom	$\longmmappedfrom$	\longmmappedfrom
$\Mmapsto$	\Mmapsto	$\Longmmapsto$	\Longmmapsto
$\Mmappedfrom$	\Mmappedfrom	$\Longmmappedfrom$	\Longmmappedfrom
$\varparallel$	\varparallel	$\varparallelinv$	\varparallelinv
$\nvarparallel$	\nvarparallel	$\nvarparallelinv$	\nvarparallelinv

Продолжение табл. 16.15

$\colonapprox$	\colonapprox	$\colonsim$	\colonsim
$\Colonapprox$	\Colonapprox	$\Colonsim$	\Colonsim
$\doteq$	\doteq	$\Doteq$	\Doteq, \doteqdot
$\multimap$	\multimap	$\multimapinv$	\multimapinv
$\multimapdot$	\multimapdot	$\multimapdotinv$	\multimapdotinv
$\multimapboth$	\multimapboth	$\multimapdotboth$	\multimapdotboth
$\multimapdotbothA$	\multimapdotbothA	$\multimapdotbothB$	\multimapdotbothB
$\VDash$	\VDash	$\VvDash$	\VvDash
$\cong$	\cong	$\preceqq$	\preceqq
$\succeqq$	\succeqq	$\nprecsim$	\nprecsim
$\nsuccsim$	\nsuccsim	$\nlesssim$	\nlesssim
$\ngtrsim$	\ngtrsim	$\nlessapprox$	\nlessapprox
$\ngtrapprox$	\ngtrapprox	$\npreccurlyeq$	\npreccurlyeq
$\nsucccurlyeq$	\nsucccurlyeq	$\ngtrless$	\ngtrless
$\nlessgtr$	\nlessgtr	$\nbumpeq$	\nbumpeq
$\nBumpeq$	\nBumpeq	$\nbacksim$	\nbacksim
$\nbacksimeq$	\nbacksimeq	$\neq$	\neq, \ne
$\nasymp$	\nasymp	$\nequiv$	\nequiv
$\nsim$	\nsim	$\napprox$	\napprox
$\nsubset$	\nsubset	$\nsupset$	\nsupset
$\llless$	\llless, \lll	$\gggtr$	\gggtr, \ggg
$\nll$	\nll	$\ngg$	\ngg
$\nthickapprox$	\nthickapprox	$\napproxeq$	\napproxeq
$\nprecapprox$	\nprecapprox	$\nsuccapprox$	\nsuccapprox
$\npreceqq$	\npreceqq	$\nsucceqq$	\nsucceqq
$\nsimeq$	\nsimeq	$\notin$	\notin
$\notni$	\notni, \notowns	$\nSubset$	\nSubset
$\nSupset$	\nSupset	$\nsqsubseteq$	\nsqsubseteq
$\nsqsupseteq$	\nsqsupseteq	$\coloneqq$	\coloneqq
$\eqqcolon$	\eqqcolon	$\coloneq$	\coloneq
$\eqcolon$	\eqcolon	$\Coloneqq$	\Coloneqq
$\Eqqcolon$	\Eqqcolon	$\Coloneq$	\Coloneq
$\Eqcolon$	\Eqcolon	$\strictif$	\strictif
$\strictfi$	\strictfi	$\strictiff$	\strictiff
$\circledless$	\circledless	$\circledgtr$	\circledgtr
$\lJoin$	\lJoin	$\rJoin$	\rJoin
$\Join$	\Join, \lrJoin	$\openJoin$	\openJoin
$\lrtimes$	\lrtimes	$\opentimes$	\opentimes
$\nsqsubset$	\nsqsubset	$\nsqsupset$	\nsqsupset

Продолжение табл. 16.15

⇠	\dashleftarrow	⇢	\dashrightarrow
⇠⇢	\dashleftrightarrow	⇜	\leftsquigarrow
↠̸	\ntwoheadrightarrow	↞̸	\ntwoheadleftarrow
⇗	\Nearrow	⇘	\Searrow
⇖	\Nwarrow	⇙	\Swarrow
∼	\leadstoext	⇝	\leadsto
□→	\boxright	←□	\boxleft
⊡→	\boxdotright	←⊡	\boxdotleft
◇→	\Diamondright	←◇	\Diamondleft
⟐→	\Diamonddotright	←⟐	\Diamonddotleft
□⇒	\boxRight	⇐□	\boxLeft
⊡⇒	\boxdotRight	⇐⊡	\boxdotLeft
◇⇒	\DiamondRight	⇐◇	\DiamondLeft
⟐⇒	\DiamonddotRight	⇐⟐	\DiamonddotLeft
○→	\circleright	←○	\circleleft
⊙→	\circleddotright	←⊙	\circleddotleft
⊕	\circledplus, \oplus	⊗	\circledtimes, \otimes
⊖	\circledminus, \ominus	⊘	\circledslash, \oslash
⊙	\circleddot, \odot	⫫	\Perp
⋓	\doublecup, \Cup	⋒	\doublecap, \Cap
⫰	\multimapbothvert	⫯	\multimapdotbothvert
⫯	\multimapdotbothAvert	⫰	\multimapdotbothBvert

Таблица 16.16

Дополнительные символы (пакеты pxfonts и txfonts)

α	\alphaup	β	\betaup	γ	\gammaup
δ	\deltaup	ϵ	\epsilonup	ε	\varepsilonup
ζ	\zetaup	η	\etaup	θ	\thetaup
ϑ	\varthetaup	ι	\iotaup	κ	\kappaup
λ	\lambdaup	μ	\muup	ν	\nuup
ξ	\xiup	π	\piup	ϖ	\varpiup
ρ	\rhoup	ϱ	\varrhoup	σ	\sigmaup
ς	\varsigmaup	τ	\tauup	υ	\upsilonup
ϕ	\phiup	φ	\varphiup	χ	\chiup
ψ	\psiup	ω	\omegaup	◇	\Diamond
⟐	\Diamonddot	◆	\Diamondblack	ƛ	\lambdaslash
ƛ	\lambdabar	♧	\varclubsuit	♦	\vardiamondsuit
♥	\varheartsuit	♤	\varspadesuit	⫪	\Top
⫫	\Bot	g	\varg	□	\Box, \square

Таблица 16.17

Большие символы (пакеты pxfonts и txfonts)

$\bignplus$	\bignplus	$\bigsqcupplus$	\bigsqcupplus
$\bigsqcapplus$	\bigsqcapplus	$\bigsqcap$	\bigsqcap
$\bigsqcap$	\bigsqcap	$\varprod$	\varprod
$\oiint$	\oiint	$\oiiint$	\oiiint
$\ointctrclockwise$	\ointctrclockwise	$\ointclockwise$	\ointclockwise
$\varointctrclockwise$	\varointctrclockwise	$\varointclockwise$	\varointclockwise
$\sqint$	\sqint	$\sqiintop$	\sqiintop
$\sqiiintop$	\sqiiintop	$\fint$	\fint
$\iint$	\iint	$\iiint$	\iiint
$\iiiint$	\iiiint	$\idotsint$	\idotsint
$\oiintctrclockwise$	\oiintctrclockwise	$\oiintclockwise$	\oiintclockwise
$\varoiintctrclockwise$	\varoiintctrclockwise	$\varoiintclockwise$	\varoiintclockwise
$\oiiintctrclockwise$	\oiiintctrclockwise	$\oiiintclockwise$	\oiiintclockwise
$\varoiiintctrclockwise$	\varoiiintctrclockwise	$\varoiiintclockwise$	\varoiiintclockwise

Таблица 16.18

Разделители (пакеты pxfonts и txfonts)

$\llbracket$ \llbracket	$\rrbracket$ \rrbracket	$\lbag$ \lbag	$\rbag$ \rbag

Таблица 16.19

Готический и контурный алфавиты (пакеты pxfonts и txfonts)

\mathfrak									\mathbb								
$\mathfrak{A}\mathfrak{a}$	$\mathfrak{B}\mathfrak{b}$	$\mathfrak{C}\mathfrak{c}$	$\mathfrak{D}\mathfrak{d}$	$\mathfrak{E}\mathfrak{e}$	$\mathfrak{F}\mathfrak{f}$	$\mathfrak{G}\mathfrak{g}$	$\mathfrak{H}\mathfrak{h}$	$\mathfrak{I}\mathfrak{i}$	$\mathbb{A}$	$\mathbb{B}$	$\mathbb{C}$	$\mathbb{D}$	$\mathbb{E}$	$\mathbb{F}$	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{I}$
$\mathfrak{J}\mathfrak{j}$	$\mathfrak{K}\mathfrak{k}$	$\mathfrak{L}\mathfrak{l}$	$\mathfrak{M}\mathfrak{m}$	$\mathfrak{N}\mathfrak{n}$	$\mathfrak{O}\mathfrak{o}$	$\mathfrak{P}\mathfrak{p}$	$\mathfrak{Q}\mathfrak{q}$	$\mathfrak{R}\mathfrak{r}$	$\mathbb{J}$	$\mathbb{K}$	$\mathbb{L}$	$\mathbb{M}$	$\mathbb{N}$	$\mathbb{O}$	$\mathbb{P}$	$\mathbb{Q}$	$\mathbb{R}$
$\mathfrak{S}\mathfrak{s}$	$\mathfrak{T}\mathfrak{t}$	$\mathfrak{U}\mathfrak{u}$	$\mathfrak{V}\mathfrak{v}$	$\mathfrak{W}\mathfrak{w}$	$\mathfrak{X}\mathfrak{x}$	$\mathfrak{Y}\mathfrak{y}$	$\mathfrak{Z}\mathfrak{z}$		$\mathbb{S}$	$\mathbb{T}$	$\mathbb{U}$	$\mathbb{V}$	$\mathbb{W}$	$\mathbb{X}$	$\mathbb{Y}$	$\mathbb{Z}$	

PSNFSS, где отсутствующие в шрифтах PostScript символы восполняются из шрифтов Computer Modern, в шрифтах Янга Рю не только имеются все символы, присутствующие в Computer Modern, но ещё добавлено много новых математических обозначений.

Производимые пакетами подмены шрифтов детализированы в табл. 16.12 и 16.13. Добавленные математические символы и соответствующие им команды перечислены в табл. 16.14–16.18, причём таблицы построены при помощи пакета txfonts.

При загрузке любого из пакетов pxfonts, txfonts буква g в математических формулах может отображаться чуточку различающимися литерами. По умолчанию g печатается как «g», но команда \varg (табл. 16.16) печатает «g». Чтобы везде в математических формулах заменить литеру «g» на литеру «g», достаточно загрузить пакет pxfonts или pxfonts с опцией varg:

```
\usepackage[varg]{pxfonts}
```

Оба пакета вводят математические алфавиты \mathfrak, \mathbb (такие же алфавиты определены в пакете amsfonts, раздел 8.3). Буквы, которые они печатают, представлены в табл. 16.19. Команда $\mathfrak{...}$ печатает готические буквы, прописные и строчные, а $\mathbb{...}$ — контурные прописные буквы.

Хотя шрифты Янга Рю не содержат русских букв, их можно использовать для подмены шрифтов в математических формулах, отменив подмену текстовых шрифтов, произведённую пакетами pxfonts и txfonts. Вот какую последовательность команд в преамбуле входного файла мы использовали, чтобы выполнить компиляцию текста с таблицами для данного раздела:

```
\usepackage{txfonts}
\renewcommand{\rmdefault}{mnt}  % Monotype New Times Roman
\renewcommand{\sfdefault}{cmss} % CM San Serif
\renewcommand{\ttdefault}{cmtt} % CM Typewriter
\usepackage[cp1251]{inputenc}
\usepackage[russian]{babel}
```

После загрузки пакета txfonts команды \sfdefault и \ttdefault были изменены так, чтобы восстановить их значения, принятые по умолчанию, а команде \rmdefault было присвоено значение, которое ей даёт пакет TimesC, рассмотренный в следующем разделе.

16.10. Пакеты FontsC

Существует несколько пакетов, которые позволяют подключать русские шрифты PostScript. Наиболее известным из них является пакет pscyr Александра Лебедева. Гипотетически этот пакет может приводить к конфликтам с другими пакетами, поскольку использует обозначения гарнитур, противоречащие схеме, принятой в системе LaTeX[11]. Поэтому мы расскажем о конкурирующей коллекции пакетов FontsC, разработанной

[11] Эта схема разработана Карлом Берри (Berry, Karl).

Игорем Котельниковым. Эти пакеты, наряду с поддержкой PostScript-шрифтов, позволяют использовать в документах LaTeX шрифты TrueType и OpenType, поставляемые корпорацией Microsoft (частично по лицензии Monotype) в составе операционных систем Windows начиная с Windows 2000[12]. PostScript-шрифты, поддерживаемые пакетами коллекции FontsC, разработаны фирмой ParaType. Некоторые из них распространяются бесплатно. Все пакеты FontsC перечислены в табл. 16.20, а в табл. 16.21 приведены образцы и названия шрифтов. Выделить шрифты PostScript в табл. 16.20 можно по принадлежности к фирме-производителю. Обозначение гарнитур PostScript-шрифтов, разработанных компанией ParaType, начинается с буквы `t`; аббревиатуры гарнитур шрифтов TrueType и OpenType компаний Microsoft и Monotype, начинаются с букв `j` и `m` соответственно.

В большей части приложений для подключения перечисленных шрифтов достаточно загрузить 1–2 пакета из коллекции FontsC. Эти пакеты подменяют стандартные шрифты Computer Modern, используемые в документах LaTeX по умолчанию. Подмена шрифтов производится посредством переопределения деклараций `\rmdefault`, `\sfdefault`, `\ttdefault` согласно табл. 16.20.

Гарнитура Arial несколько крупнее, чем другие гарнитуры при номинально равных размерах. Как следствие, смешивание Arial с другими гарнитурами в одной строке делает текст эстетически малопривлекательным. Проблему решает загрузка пакета ArialC с опцией `[scaled=`*scale*`]`. Например,

```
\usepackage[scaled=0.95]{ArialC}
```

уменьшает гарнитуру Arial до 95% от номинального размера. Параметр *scale* (вместе со знаком =) можно опустить, что соответствует масштабу 92%. Пакет BookManualC также принимает опцию `[scaled=`*scale*`]`, причём она действует только на гарнитуру Arial — одну их трёх, загружаемых этим пакетом.

Пакет mathmnt заменяет шрифты в формулах. Пакет TimesC имеет две опции: `nomath` и `math`. По умолчанию используется первая, а шрифты заменяются только в тексте. Если же пакет загружен с опцией `math`, происходит также частичная подмена математических шрифтов аналогично тому, как это делает пакет mathmnt.

Вместо загрузки пакета можно просто переопределить декларации `\rmdefault`, `\sfdefault`, `\ttdefault` в преамбуле файла с исходным текстом документа. Следующий пример показывает, как подключить гарнитуры Bookman Old Style и Century Gothic соответственно вместо CM Roman (шрифт с засечками) и CM Sans Serif (шрифт без засечек), используемых в документе LaTeX по умолчанию:

```
\renewcommand{\rmdefault}{mbk}
\renewcommand{\sfdefault}{myg}
```

Гарнитуры Bookman Old Style и Century Gothic, как и вообще все гарнитуры OpenType, впервые включённые в состав операционных систем Windows начиная с Windows 2000, не загружаются ни одним из пакетов, перечисленных в табл. 16.20. Число шрифтов всё

[12] Гарнитуры Arial, Courier New и Times New Roman поставляются также с Windows 95 и 98, а гарнитура Garamond — с Windows 98.

Таблица 16.20

Шрифты, поддерживаемые пакетами FontsC версии 1.1.
Пропуск означает, что гарнитура не изменяется пакетом

Пакет	\rmdefault	\sfdefault	\ttdefault	Формулы
	cmr	cmss	cmtt	≈ CM Roman
AcademyC	tdy			
AvantC		tag		
BalticaC	tl5			
BaskerC	tnb			
ChanceryC	tzc			
LiteraturnayaC	tl6			
mathmnt				≈ Times New Roman
OffSerifC			to8	
PetersburgC	tp7			
SclBookC	tcs	tag	to8	
StScriptC	tud			
GaramondC	mgm	ma1	mcr	
TimesC	mnt	ma1	mcr	≈ Times New Roman
VerdanaC	jvn			

Таблица 16.21

Шрифты, поддерживаемые пакетами FontsC версии 1.1

ENC	*family*	series	*shape*	Гарнитура
OT1 T1 T2A	jvn	m bx	n sl	Verdana
OT1 T1 T2A	ma1	m bx mc bc	n sl sc	Arial
OT1 T1 T2A	maq	m bx mc bc	n sl sc	Book Antiqua
OT1 T1 T2A	mbk	m bx	n sl sc	Bookman Old Style
OT1 T1 T2A	mcr	m bx	n sl sc	Courier New
OT1 T1 T2A	mgm	m bx	n it sl sc	Garamond
OT1 T1 T2A	mnt	m bx	n it sl sc ui	Times New Roman
OT1 T1 T2A	myg	m bx	n sl sc	Century Gothic
OT1 T1 T2A	tag	m bx	n sl sc	Avant Garde Gothic
OT1 T1 T2A	tcs	m bx	n it sl sc ui	School Book
OT1 T1 T2A	tdy	m bx	n it sl sc ui	Академическая
OT1 T1 T2A	tl5	m bx	n it sl sc ui	Балтика
OT1 T1 T2A	tl6	m bx	n it sl sc ui	Литературная
OT1 T1 T2A	tnb	m bx	n it sl sc ui	New Baskerville
OT1 T1 T2A	to8	m bx	n sl sc	Officina Serif
OT1 T1 T2A	tp7	m bx	n it sl sc ui	Петербург
OT1 T1 T2A	tud	m	it	*Studio Script*
OT1 T1 T2A	tzc	m	it	*ZapfChancery*

время увеличивается, поэтому дальнейшее размножение пакетов в настоящее время вряд ли оправдано.

При загрузке пакетов происходит подмена шрифтов во всём печатном документе. Чтобы выделить отдельный фрагмент текста, можно использовать команды, описанные в разделе 16.3:

```
...не путать с
 {\usefont{T2A}{tud}{m}{it} Рукописным}!
```

. . . не путать с *Рукописным*!

При частом использовании подобных выделений удобно ввести специальную команду, которая бы производила переключение шрифтов. В частности, последний пример можно было бы переделать примерно следующим образом:

```
\newcommand{\script}[1]
 {{\usefont{T2A}{tud}{m}{it}#1}}
 ...
не путать с \script{Рукописным}!
```

. . . не путать с *Рукописным*!

Я слышал отрывки — автор велик!
В нём слышится то Дант, то Шекспир...
И. Гончаров. Обломов

Глава 17

Полоса набора

Итак, большой труд почти завершён. Читатель переквалифицировался в Писателя. У него в руках рукопись (о, простите, макет!) его собственной книги, и он на пороге издательства. «Отлично! — потирает руки Издатель.— Мы давно мечтаем от таком издании. Только вот тут мы делаем *отбивку*, а у Вас...» Непонятные слова можно пропустить мимо ушей, но сделать некоторые изменения придётся. В этой главе мы расскажем о том, что определяет стиль страницы и что профессиональные редакторы называют *макетом полосы набора.* «Вот с этого и надо было начинать!» — скажет Читатель. Возможно. Однако как мы ни старались подобрать для рассказа о макете полосы набора местечко где-нибудь поближе к началу книги, так ничего и не придумали, ведь чтобы создать технически совершенный печатный документ, нужно знать почти всё, что касается издательского дела.

17.1. Из чего состоит страница

Полоса набора, а проще говоря, *страница* в печатном документе состоит из трёх частей: *верхнего колонтитула*, *тела* страницы и *нижнего колонтитула*. В тело страницы входит всё, что находится между колонтитулами: текст, подстрочные примечания, рисунки и таблицы. Класс печатного документа определяет размеры всех частей страницы, а также содержание колонтитулов. Левые и правые страницы могут иметь разные размеры и разные колонтитулы, если документ подготовлен для печати на двух сторонах листа бумаги. При двусторонней печати чётные страницы располагаются на левой половине разворота книги, а нечётные — на правой. При односторонней печати все страницы считаются правыми.

По умолчанию для печатного документа класса `article`, `proc`, `report`, `slides` и `letter` устанавливается односторонняя печать, для класса `book` — двусторонняя. Опция `twoside` в декларации `\documentclass` поможет установить режим двусторонней печати также для статей и отчётов, но в письмах, слайдах и научных докладах всегда используется односторонний формат страницы.

Информация в колонтитулах, которая чаще всего включает в себя номера страниц, а иногда и названия текущей главы и раздела, призвана помочь читателю найти нужное место в печатном документе. По-видимому, не стоит дета-

лизировать, что конкретно каждый из стандартных классов заносит в колонтитулы. Немного поэкспериментировав, Читатель легко сможет получить нужную ему информацию. Ограничимся замечанием, что верхний колонтитул по умолчанию не печатается на первой странице печатного документа, на первой странице каждой главы, а также в статьях и письмах, где он только бы отвлекал внимание.

Стиль страницы (полосы набора), установленный классом печатного документа, изменяют декларации

⚠ `\pagestyle{`*page-style*`}`
⚠ `\thispagestyle{`*page-style*`}`

Область действия декларации `\pagestyle` подчиняется обычным правилам с одним уточнением: она начинает действовать с текущей страницы. Декларация `\thispagestyle` аналогична `\pagestyle`, но воздействует только на текущую страницу. Аргумент *page-style* может принимать четыре значения:

`plain` печатает номера страниц в нижнем колонтитуле, а верхний колонтитул пуст. Стандартные классы, кроме `book`, используют стиль `plain` по умолчанию;

`empty` означает, что верхний и нижний колонтитулы пусты. LaTeX поддерживает нумерацию страниц, но номера не печатает;

`headings` печатает номера страниц и другую информацию, определяемую классом документа (обычно заголовки разделов), в верхнем колонтитуле. Нижний колонтитул пуст. Печатный документ класса `book` использует стиль `headings` по умолчанию;

`myheadings` означает, что содержание верхнего колонтитула должен задать сам пользователь, используя команды `\markboth` и `\markright`, описанные ниже. Нижний колонтитул пуст.

Поскольку LaTeX определяет содержание колонтитулов в конце форматирования страницы, декларацию `\pagestyle` обычно вставляют после команд (типа `\chapter`), которые начинают печатать текст с новой страницы.

Декларации

⚠ `\markboth{`*left*`}{`*right*`}`
⚠ `\markright{`*right*`}`

имеют смысл при выборе стиля страницы `headings` или `myheadings`. В первом случае (`headings`) они переопределяют содержание верхнего колонтитула, установленное классом печатного документа (посредством этих же деклараций). Во втором случае (`myheadings`) использование `\markboth` или `\markright` почти обязательно. Аргументы *left* и *right* должны содержать ту информацию, которая пойдёт соответственно на левую и правую страницы, причём при односторонней печати все страницы считаются правыми. В верхний колонтитул левой страницы переносится аргумент *left* последней команды `\markboth` перед концом страницы. В верхний колонтитул правой страницы переносится аргумент *right* первой

декларации `\markright` или `\markboth` на странице, а если их нет, то последней из этих деклараций перед началом страницы.

Как *left*, так и *right* обрабатываются в строковой моде. Они являются подвижными аргументами, поэтому хрупкие команды в них должны быть защищены командой `\protect`.

При выборе стиля страницы `headings` класс печатного документа устанавливает, какие команды секционирования и при помощи какой из команд `\markboth`, `\markright` заносят название раздела в колонтитулы:

Стиль печати	Класс документа		Декларация
	`book`, `report`	`article`, `proc`	
Двусторонний	`\chapter`	`\section`	`\markboth`[a]
	`\section`	`\subsection`	`\markright`
Односторонний	`\chapter`	`\section`	`\markright`

[a] Устанавливает пустой верхний колонтитул для правой страницы.

Чтобы переопределить эти установки, необходимо поместить `\markboth` сразу же после команды секционирования, а `\markright` непосредственно перед и после команды секционирования, но первая команда должна быть опущена, если команда секционирования начинает новую страницу. Типичный случай использования команд `\markboth` и `\markright` связан с применением `*`-формы команд секционирования. Например, если требуется, чтобы очередная глава не имела номера, её следует начать с команды `\chapter*`. Однако `\chapter*` не только не нумерует главу, но также не заносит её название в верхние колонтитулы, как это делает обычная форма команды `\chapter`. Чтобы исправить этот недочёт, как раз необходимо применить команду `\markboth`:

```
\chapter*{Вместо предисловия}
\markboth{Вместо предисловия}{Вместо предисловия}
```

Верхний колонтитул первой страницы печатного документа всегда пуст. Если Читателя это не устраивает, то проще всего создать пустой титульный лист процедурой `titlepage`.

Специальные декларации для переопределения содержания нижнего колонтитула не предусмотрены. Нижний колонтитул можно изменить с помощью пакета fancyheadings. Мы его не описываем, поскольку являемся сторонниками классического подхода к оформлению печатных изданий.

Декларация

⚠ `\pagenumbering{`*num-style*`}`

присваивает счётчику страниц `page` значение 1 и определяет стиль нумерации страниц. Она имеет глобальную область действия. Аргумент *num-style* должен совпадать с именем одной из команд (за исключением `\fnsymbol`), которые печатают значения счётчиков (раздел 2.9), но не иметь обратного слеша:

`arabic` для печати номеров страниц арабскими цифрами,

`roman` строчными римскими цифрами,

`Roman` прописными римскими цифрами,

`alph` строчными латинскими буквами,

`Alph` прописными латинскими буквами,

`asbuk` строчными русскими буквами,

`Asbuk` прописными русскими буквами.

По умолчанию нумерация страниц ведётся арабскими цифрами. Печать номеров страниц русскими буквами возможна при загрузке пакета babel с опцией `russian`, поскольку команды `\asbuk` и `\Asbuk` определены в этом пакете.

Декларация `\pagenumbering{roman}` вводит нумерацию страниц римскими цифрами, а `\pagenumbering{arabic}` восстанавливает арабские цифры. Каждое использование декларации `\pagenumbering` начинает отсчёт страниц с единицы, присваивая первый номер текущей странице. Таким образом, чтобы начать предисловие к книге со страницы с номером в виде римской единицы, достаточно поставить `\pagenumbering{roman}` где-нибудь в преамбуле, а сразу после начала первой главы (после первой команды `\chapter`) — декларацию `\pagenumbering {arabic}`. Тогда отсчёт страниц в основном тексте начнётся заново, и они будут пронумерованы арабскими цифрами. Впрочем, в классе book все такие изменения нумерации страниц делают декларации `\frontmatter` и `\mainmatter` (раздел 15.2), но делают с помощью `\pagenumbering`. На всякий случай напомним, что при нумерации страниц буквами номер последней страницы не должен превышать числа букв в алфавите.

17.2. Настройка макета

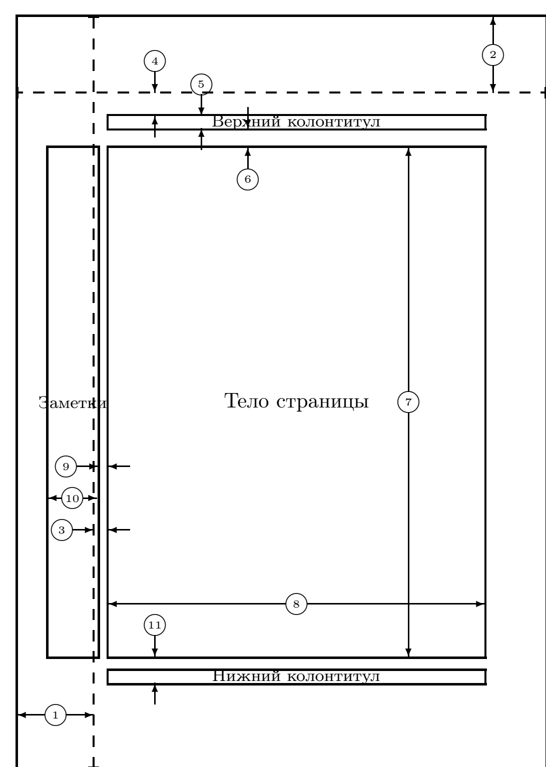
На рисунке 17.1 показан макет полосы набора данной книги, построенный командой

`\layout` (layout)

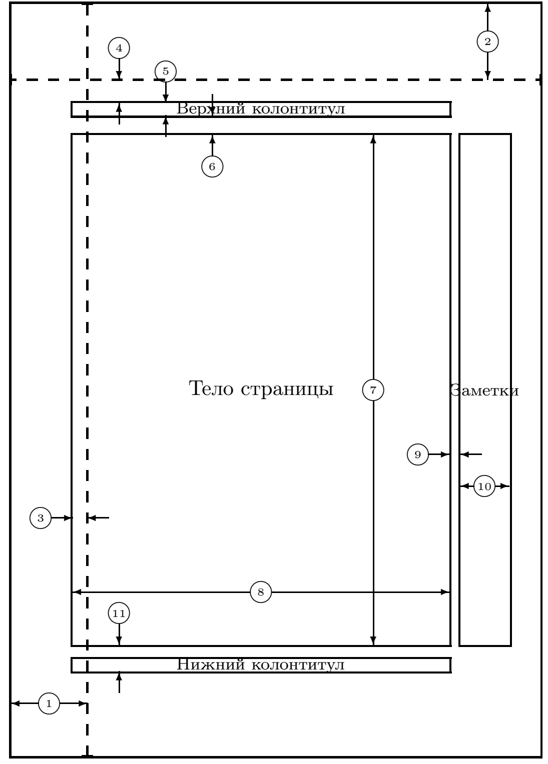
из одноимённого пакета Кента Макферсона (McPherson, Kent), входящего в коллекцию tools. Эта команда крайне полезна при настройке макета страницы[1]. При двусторонней печати команда `\layout` изображает макет двух страниц: чётной (левой) и нечётной (правой), как на рис. 17.1. При односторонней печати отображается только одна страница.

Стандартные классы печатных документов вычисляют геометрические размеры полосы набора, основываясь на размерах листа бумаги, на котором планируется печатать документ. Ширина и высота листа хранятся в командах

[1] Пакет layout локализован для некоторых языков, но не для русского. Его можно загрузить с опциями `dutch`, `english`, `german`, `italian` и т. д. Мы переопределили команды `\Headertext`, `\Bodytext`, `\Footertext`, `\MarginNotestext`, `\oneinchtext`, `\notshown`, чтобы получить надписи на русском языке на рис. 17.1.



1	1 дюйм + \hoffset	2	1 дюйм + \voffset
3	\evensidemargin = 14pt	4	\topmargin = 22pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 482pt	8	\textwidth = 356pt
9	\marginparsep = 10pt	10	\marginparwidth = 47pt
11	\footskip = 25pt		\marginparpush = 5pt (не показано)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 500pt		\paperheight = 711pt



1	1 дюйм + \hoffset	2	1 дюйм + \voffset
3	\oddsidemargin = -14pt	4	\topmargin = 22pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 482pt	8	\textwidth = 356pt
9	\marginparsep = 10pt	10	\marginparwidth = 47pt
11	\footskip = 25pt		\marginparpush = 5pt (не показано)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 500pt		\paperheight = 711pt

Рис. 17.1. Макет полосы набора данной книги

```
\paperwidth
\paperheight
```

соответственно. При размещении изображения на листе предусмотрены поля шириной 1 дюйм, т. е. 2,54 см от верхней и нижней кромки листа. Эти поля показаны на рис. 17.1 пунктирной линией. Величину полей можно увеличить или уменьшить, изменив значения командных длин

```
\hoffset
\voffset
```

однако делать это не рекомендуется[2]. Значения `\hoffset` и `\voffset`, по умолчанию равные нулю, прибавляются к стандартной величине полей, т. е. к 1 дюйму, соответственно, слева и сверху.

Значения параметров `\paperheight` и `\paperwidth` зависят от заказанного формата листа бумаги и по умолчанию соответствуют формату `letterpaper`. Чтобы заказать лист формата A4, необходимо в `\documentclass` указать опцию `a4paper`. Полный набор поддерживаемых форматов бумаги и соответствующие им опции перечислены в разделе 3.2. Изменение значений `\paperheight` и `\paperwidth` в преамбуле входного файла не имеет никакого эффекта, так как параметры, определяющие геометрические размеры полосы набора, к тому времени уже вычислены. Зато вычисленные размеры можно изменить. Обычно изменения вносят именно в преамбуле документа, так изменение геометрических размеров страницы где-нибудь в середине входного файла может привести к непредсказуемым последствиям. Ниже мы приводим расшифровку всех геометрических параметров, в том числе и тех, которые не показаны на рис. 17.1.

`\textheight` — высота текста, т. е. вертикальный размер тела страницы (без колонтитулов).

`\textwidth` — ширина текста, т. е. горизонтальный размер колонтитулов и тела страницы. Переопределяется внутри процедур, изменяющих правую и/или левую границы текста (раздел 5.4.4).

`\oddsidemargin` — расстояние между левым краем листа и левым краем текста на правой странице минус 1 дюйм.

`\evensidemargin` — то же самое для левосторонней страницы.

`\topmargin` — расстояние между верхним краем листа и верхним колонтитулом минус 1 дюйм.

`\headheight` — высота верхнего колонтитула.

`\headsep` — вертикальное расстояние между верхним колонтитулом и телом текстовой страницы.

---

[2] Некоторые пользователи изменяют величину полей, пытаясь скомпенсировать различия в настройках принтеров. Лучше всё-таки правильно настроить принтер.

\topskip — минимальное расстояние между верхним краем тела страницы и нижним краем (т. е. базисной линией) первой строки текста. Соответствует \baselineskip (см. ниже) для первой строки страницы.

\footheight — высота нижнего колонтитула.

\footskip — вертикальное расстояние между нижним краем текста и нижним краем нижнего колонтитула; измеряется между базисными линиями последних строк текста нижнего колонтитула.

\linewidth — ширина строки; равна значению \textwidth за исключением строк внутри процедур форматирования абзацев, таких как quote или itemize (глава 4). Значение \linewidth не должно изменяться командами, изменяющими длину.

\parindent — ширина отступа в начале абзаца. Внутри парбокса (раздел 9.2) устанавливается равной нулю. Может быть изменена в любом месте.

\baselineskip — минимальное расстояние между нижними краями (базисными линиями) последовательных строк текста. Расстояние между некоторыми строками может быть больше, если они содержат высокие объекты. Значение \baselineskip устанавливается декларациями, изменяющими размер шрифта (см. табл. 1.2).

Значение \baselineskip, используемое для всего абзаца, определяется значением, действовавшим на момент завершения абзаца пустой строкой, командой \par или одной из процедур, печатающих текст с абзацного отступа или отдельной строкой. Значение \baselineskip может быть изменено в любом месте входного файла. Если на странице размещается один абзац, то величина \textheight, делённая на \baselineskip, равна числу строк на странице.

\baselinestretch — десятичное число, равное величине интерлиньяжа (межстрочного интервала); по умолчанию равно 1. Значение \baselinestretch изменяется декларацией \renewcommand. Например, команда \renewcommand {\baselinestretch}{1.25} в преамбуле входного файла приведёт к тому, что весь печатный документ будет напечатан через 1,5 интервала (как следующий абзац). Чтобы напечатать документ примерно через два интервала, следует задать значение \baselinestretch, равное 1.67. Половине интервала соответствует значение 0.75.

Действующее значение расстояния между строчками \baselineskip равно умноженному на \baselinestretch стандартному значению, которое приписано каждому размеру шрифта. Величина межстрочного интервала первоначально устанавливается командой \begin{document}. Дальнейшие изменения действующего межстрочного интервала осуществляются декларациями переключения размера шрифта[3] (см. табл. 16.6), так что само по себе пере-

[3] Изменение интервала между строками возможно также посредством команды \linespread, описанной в разделе 16.3.

определение \baselinestretch ещё не изменяет межстрочный интервал. Все изменения вступают в действие в момент изменения размера шрифта.

Любое изменение межстрочного интервала должно быть частью модификации стиля печатного документа в целом. Квалифицированное изменение стиля печатного документа под силу только опытному типографскому дизайнеру.

\parskip — дополнительный вертикальный пробел, вставляемый перед абзацем.

\marginparpush — минимальное вертикальное расстояние на между двумя последовательными заметками на полях.

\marginparsep — горизонтальное расстояние между внешним краем текста и заметкой на полях.

\marginparwidth — ширина заметок на полях.

\columnsep — расстояние между колонками при печати в две колонки.

\columnseprule — ширина вертикальной линии между колонками при печати в две колонки. Значение по умолчанию равно нулю, что соответствует невидимой линии.

Параметры настройки, характеризующие горизонтальные размеры, суть нерастяжимые длины. Вертикальные размеры обычно растяжимы; однако расстояние между строками лучше всего выбирать фиксированным.

Значение перечисленных параметров настройки можно изменить посредством \setlength. Например, если в преамбуле стоит

```
\setlength{\columnseprule}{0.4pt}
```

то при печати в две колонки на каждой странице печатного документа между колонками будет проведена черта толщиной 0,4 pt.

17.3. Печать в две колонки

Как уже упоминалось в главе 3, LaTeX сформатирует печатный документ в две колонки, если в списке опций \documentclass указать twocolumn. Опцию twocolumn следует использовать, если необходимо напечатать в две колонки бо́льшую часть текста. В одноколоночном документе переход к печати в две колонки осуществляет команда

⚠ \twocolumn[*text*]

Она начинает новую страницу (используя команду \clearpage) и начинает печать следующего за ней текста в две колонки. Если имеется необязательный аргумент *text*, он будет напечатан в верхней части новой страницы в одну колонку; если опция отсутствует, то печать в две колонки начинается от начала страницы. Возврат к одноколоночному формату осуществляет команда

⚠ `\onecolumn`

Она также исполняет команду `\clearpage` и продолжает печатать текст с новой страницы. Более гибкого форматирования печатного документа позволяет достичь пакет multicol из коллекции tools (раздел 17.4).

Класс `article`, загруженный с опцией `twocolumn`, ориентирован на форматирование текста в две колонки, главным образом, для публикации журнальных статей. В двухколоночном формате иногда также публикуются труды конференций. Издатели трудов часто предъявляют особые требования к оформлению публикуемых материалов, отличающиеся от оформления книг. В этом случае вместо `article` лучше выбрать класс `proc`, по умолчанию форматирующий текст в две колонки (более того, печать в одну колонку печатного документа класса `proc` невозможна).

При печати в две колонки подстрочные примечания размещаются под соответствующей колонкой. Пакет ftnright из коллекции tools, написанный Франком Миттельбахом (Mittelbach, Frank) размещает все подстрочные примечания под правой колонкой.

Процедуры `figure` и `table`, предназначенные для размещения рисунков и таблиц, выделяют для них место в одной колонке вне зависимости от фактической ширины рисунка или таблицы. Для размещения широких плавающих объектов следуют использовать `*`-форму этих процедур, которая резервирует место сразу в двух колонках. При наличии в одном документе и одноколоночных, и двухколоночных рисунков (или таблиц) порядок их следования может нарушаться, так что двухколоночный рисунок может оказаться впереди следующего за ним одноколоночного рисунка (или наоборот). Этот изъян в алгоритме размещения плавающих объектов устраняет загрузка пакета fixltx2e. Пакет fixltx2e лечит ещё несколько дефектов, которые, вероятно, будут исправлены в новых выпусках системы LaTeX. В частности, он разрешает редкую проблему, когда в колонтитул попадает текст из второй колонки, а не из первой, что кажется более логичным.

17.4. Пакет multicol

Описанными выше средствами невозможно сформировать страницу с переменным числом колонок, так как команды `\twocolumn` и `\onecolumn` всегда начинают печатать текст с новой страницы. Кроме того, колонки оказываются несбалансированными, т. е. имеют разную длину. Пакет multicol, написанный Франком Миттельбахом (Mittelbach, Frank), решает эти проблемы. Он вводит процедуру `multicols`, которая

- может создавать до 10-ти колонок на произвольном числе страниц;
- выравнивает колонки на последней странице так, чтобы их длины были приблизительно одинаковы;

- может использоваться внутри другой процедуры, такой как `figure` или `minipage`, где она производит бокс с текстом, распределённым на заданное число колонок;
- может проводить между колонками вертикальные разделительные линии заданной толщины;
- имеет удобные средства глобальной и индивидуальной регулировки формата колонок.

17.4.1. Процедура multicols

```
\begin{multicols}{number}[preface][skip]
    text
\end{multicols}
```

(multicol)

В простейшем варианте процедуры `multicols` достаточно указать число колонок *number*. Тело процедуры *text* может содержать произвольный текст и любые команды LaTeX'а, кроме плавающих объектов и заметок на полях (глава 11). Нередко многоколоночному тексту требуется предпослать некое общее предисловие. Его надо поместить в необязательный аргумент *preface*. Тогда LaTeX постарается разместить *preface* и начало текста в колонках на одной странице.

```
\begin{multicols}{2}
   [\section*{Простой совет}]
Это текст, распределённый между
двумя колонками. Если колонки
слишком узкие, полезно отменить
выравнивание по правой границе.
\end{multicols}
```

Простой совет

Это текст, распределённый между двумя колонками. Если колонки слишком узкие, полезно отменить выравнивание по правой границе.

Процедура `multicols` начинает новую страницу, если на текущей странице недостаточно свободного места. Предельный размер свободного места контролирует некий глобальный параметр. При наличии предисловия *preface* этот размер может оказаться слишком малым. В таком случае можно либо изменить его глобальное значение (см. ниже), либо подобрать нужное значение только для текущей процедуры, указав его во втором необязательном аргументе *skip*. Например, после

```
\begin{multicols}{2}
   [\section*{Простой совет}]
   [7cm]
Это текст...
```

новая страница будет начата, если на текущей осталось менее 7 см свободного пространства.

17.4.2. Настройка процедуры `multicols`

```
\premulticols
\postmulticols
\multicolsep
\columnsep
\columnseprule
```

(multicol)

Форматируя текст, процедура `multicols` распознаёт значения нескольких параметров.

Прежде всего она измеряет высоту свободного пространства на странице, чтобы определить, достаточно ли оно для размещения некоторой порции текста в колонках. Минимально требуемый размер задаётся опцией *skip*, а при её отсутствии — командной длиной `\premulticols`, которая может быть переопределена стандартными командами LaTeX'а, предназначенными для работы с длинами (раздел 2.10). По умолчанию `\premulticols` составляет 50 pt. Если имеющееся пространство меньше `\premulticols`, происходит переход на новую страницу. В конце выполнения процедуры остающееся на странице место сравнивается со значением `\postmulticols`, равным по умолчанию 20 pt. Перед многоколоночным текстом и после него вставляется растяжимая длина `\multicolsep`, в естественном состоянии равная 12 pt.

Расстояние между колонками и толщина разделительной вертикальной линии определяются теми же командами `\columnsep` и `\columnseprule`, что и в обычном двухколоночном формате (раздел 17.3).

```
\setlength{\columnseprule}{.4pt}
\begin{multicols}{3}\raggedright
 Это текст, распределённый между
 тремя колонками. Здесь отменено
 выравнивание по правой границе.
\end{multicols}
```

Это текст, распреде-лённый между	тремя колонками. Здесь отменено	выравнива-ние по правой границе.

17.4.3. Балансировка колонок

```
\flushcolumns
\raggedcolumns
```

(multicol)

По умолчанию, когда действует декларация `\flushcolumns`, процедура пытается сделать все колонки одинаковой длины за счёт растяжения существующих в колонках вертикальных пробелов. Если во входной файл введена декларация `\raggedcolumns`, избыточные пробелы собираются на дне колонок.

```
unbalance
columnbadness
finalcolumnbadness
tracingmulticols
```

(multicol)

В конце выполнения процедуры остающийся текст балансируется, чтобы создать колонки равной длины. Если желательно поместить в левые колонки больше текста, чем в самую правую, следует увеличить значение счётчика `unbalance`, по умолчанию равное нулю. Этот счётчик определяет количество дополнительных (пустых) строк, которое должно быть вставлено в последнюю колонку.

```
\setlength{\columnseprule}{.4pt}
\begin{multicols}{3}\raggedright
 Это текст, распределённый между
 тремя колонками. Здесь отменено
 выравнивание по правой границе.
 \setcounter{unbalance}{1}
\end{multicols}
```

Это текст, распреде- лённый между тремя	колонками. Здесь отменено выравнива- ние по	правой границе.

Балансировка колонок управляется ещё двумя счётчиками `finalcolumnbadness` и `columnbadness`. Когда LaTeX конструирует боксы (такие как колонки), он вычисляет специальный параметр, характеризующий качество бокса. Для хорошего бокса этот параметр равен 0, для плохого — 10 000, а для переполненного — 100 000. Балансировочный алгоритм сравнивает параметр качества для возможных решений по распределению текста между колонками. Если для любой колонки, кроме последней, величина параметра превышает `columnbadness`, решение бракуется. По умолчанию `columnbadness` = 10 001. Когда алгоритм находит подходящее решение, он анализирует качество последней колонки. Если оказывается, что её параметр качества больше, чем `finalcolumnbadness`, к колонке добавляется вертикальный пробел снизу, и она выходит короче других. По умолчанию `finalcolumnbadness` = 99 999.

Действия алгоритма можно проследить, задав счётчику `tracingmulticols` значение больше нуля. Чем оно больше, тем более подробную информацию о своих действиях алгоритм балансировки выводит в файле протокола. Той же цели служат опции `errorshow`, `infoshow`, `balancingshow`, `markshow` и `debugshow` при загрузке пакета командой `\usepackage`. Каждая из них просто задаёт значение счётчика `tracingmulticols` на единицу больше, чем предшествующая ей в нашем списке.

17.4.4. Сноски и плавающие объекты

В `multicols` можно использовать процедуры `figure*` и `table*`, которые размещают плавающие объекты на полную ширину страницы. Команда `\marginpar` для заметок на полях и обычные процедуры `figure`, `table`, создающие плавающие фигуры в колонке, не допустимы. Ещё одно ограничение состоит в том, что плавающий рисунок или таблица никогда не попадают на страницу, где они появляются во входном файле. Другими словами, можно повлиять на их размещение, применив спецификаторы `t`, `b` и/или `p` в необязательном аргументе процедур `figure*` и `table*` (раздел 11.1), но спецификатор `h` не работает, потому что ближайшее разрешённое место есть верх следующей страницы.

Подстрочное примечание также печатается на полную ширину страницы, а не под соответствующей колонкой. При определённых обстоятельствах подстрочное примечание может оказаться не на той странице, где находится текст, к которому оно относится. В этом случае пакет multicol выдаёт предупреждение

`LaTeX Warning: I moved some lines to the next page`[4].

Тогда следует проверить страницу, вызвавшую предупреждение, на предмет правильности расположения подстрочного примечания и его метки в тексте. Если метка и текст подстрочного примечания действительно оказались на разных страницах, преодолеть возникшую проблему можно, вставив в подходящее место входного файла команду `\pagebreak` (раздел 4.7).

Другой способ реагирования на такое предупреждение состоит в том, чтобы изменить значение счётчика

`collectmore` (multicol)

по умолчанию равное нулю. Если этому счётчику присвоить значение n, то алгоритм балансировки учтёт на n строк больше (или меньше, если $n < 0$) перед тем, как принять окончательное решение. Поэтому `\setcounter{collectmore}{-1}` может решить все проблемы с размещением подстрочного примечания, правда, за счёт менее оптимального подбора размеров колонок.

17.5. Брошюровка макета

Даже в самых маленьких типографиях книги печатают на больших листах бумаги, размещая несколько страниц на одном листе так, чтобы при сгибании листа по границам страниц получался буклет. Обрезав края буклета с трёх сторон, получают тетрадь. Из нескольких тетрадей сшивают книгу.

Очевидно, что для получения правильного расположения страниц в буклете необходимо изменить порядок следования страниц, ориентацию и размещение страниц на листе. Все эти операции можно выполнить с помощью одной программы `pstops`, написанной Ангусом Даггеном (Duggan, Angus). Она входит в комплект поставки большинства реализаций системы LaTeX. Мы ограничимся решением одной конкретной задачи, рассчитывая, что Читатель при необходимости сможет найти дополнительные сведения по работе с программой `pstops` в прилагаемой к ней документации.

Предположим, что имеется печатный документ, подготовленный для печати на листах бумаги размером A5. Иными словами, при компиляции документа была использована опция `a5paper`:

```
\documentclass[a5paper]{article}
```

[4] Предупреждение LaTeX'а: я переставил некоторые строчки на следующую страницу.

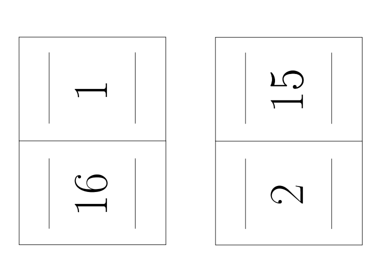


Рис. 17.2. Первые листы буклета из 16 страниц при печати по 2 страницы на лист. Цифры соответствуют номерам страниц

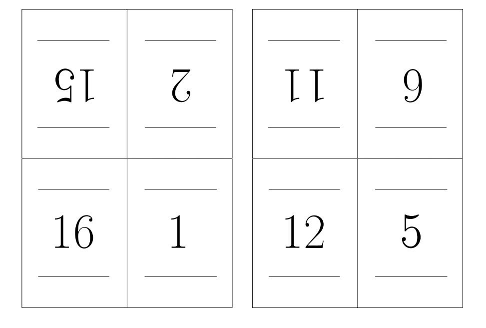


Рис. 17.3. Первые листы буклета из 16 страниц при печати по 4 страницы на лист. Цифры соответствуют номерам страниц

Предположим также, что входной файл с исходным текстом документа называется `note.tex`.

Чтобы иметь возможность производить манипуляции со страницами в электронном виде, документ необходимо преобразовать в формат PostScript в соответствии со схемой 1.6, обсуждавшейся в конце первой главы. Эта операция выполняется в два шага. Сначала производится компиляция исходного текста в формат DVI:

```
latex note
```

Затем полученный `dvi`-файл `note.dvi` преобразуется в `ps`-файл с помощью программы `dvips`:

```
dvips -ta5 note
```

Здесь нужно обратить внимание на наличие опции `-ta5`, которая показывает, что документ PostScript предназначен для печати на листах бумаги размером A5, а не A4, как предполагает программа `dvips` по умолчанию. В результате будет получен файл `note.ps`. Его можно просмотреть на экране компьютера с помощью программы GSview (рис. 1.7). С этим файлом уже можно производить операции по упорядочиванию страниц. В результате следующей команды будет получен файл с именем `a4_4.ps`, который предназначен для печати одновременно 2-х страниц на листе бумаги размером A4:

```
pstops -pa5 -d1 4:0L(1h,1w)+-3L(1h,0),-2L(1h,1w)+1L(1h,0) note.ps a4_4.ps
```

Его первые две страницы в уменьшенном виде изображены на рис. 17.2. Затем следует напечатать нечётные страницы полученного файла `a4_4` (большинство принтеров предоставляют такую возможность), перевернуть стопку отпечатанных страниц и напечатать чётные страницы. Если теперь перегнуть стопку пополам, получится готовая тетрадь с правильным порядком страниц.

Синтаксис командной строки программы pstops довольно сложен, но мы попытаемся его пояснить на приведённом примере. Ключ -pa5 указывает, что размер страницы в исходном файле note.ps соответствует листу бумаги размера A5. Можно было бы явно указать ширину и высоту листа с помощью ключей -w148mm -h210mm. Ключ -d1 указывает, что границы страниц на листе нужно обвести линией толщиной 1pt. Мы сделали это для большей наглядности. Далее следует спецификация блока страниц. Она начинается с числа страниц в каждом блоке. В данном случае это число равно 4. За ним через двоеточие следуют спецификации отдельных страниц, которые могут быть разделены знаком + или запятой. Знак + означает, что страница подлежит размещению на текущем листе бумаги, тогда как запятая вызывает переход на следующий лист. Новый блок всегда начинается с нового листа, но каждый блок может размещаться на нескольких листах, как в данном примере, где первая пара страницы в блоке отделена от второй пары запятой.

Спецификация первой страницы 0L(1h,1w) расшифровывается следующим образом. Буква L означает, что страницу нужно повернуть влево (т. е. против часовой стрелки) на 90 градусов. В скобках указаны координаты нижнего левого угла страницы (до её поворота) относительно нижнего левого угла листа в единицах ширины (w) и высоты (h) листа бумаги с размером, указанным в значении ключа -p, т. е. A5. Можно было бы указать координаты в миллиметрах или иных единицах длины. Конкретно (1h,1w) означает, что страница будет смещена на 210 миллиметров (это высота листа бумаги размером A5 и одновременно ширина листа размером A4) по горизонтали и на 148 миллиметров (это ширина листа бумаги размером A5 и половина высоты листа размером A4) по вертикали. После поворота влево на 90 градусов первая страница займёт верхнюю половину листа A4. Спецификации других страниц в блоке сконструированы по тому же принципу, поэтому каждая вторая страница в блоке, координаты которой заданы в виде (1h,0w), займёт нижнюю половину листа.

Наконец, цифры перед указателем поворота L определяют порядковый номер страницы, которая привязывается к координатам, указанным в скобках. Отсчёт порядкового номера начинается с нуля, а последний номер в блоке должен быть на единицу меньше числа страниц в блоке, т. е. в данном случае может иметь значения 0, 1, 2, 3. Минус перед номером страницы в блоке означает, что нужно взять страницу из симметричного блока от конца документа. Например, если в документе 16 страниц, то номеру 0 будут соответствовать страницы 1, 5, 9, 13; а номеру -3 отвечают страницы 16, 12, 8, 4. Таким образом, на чётные листы попадут пары страниц: [16, 1]; [12, 5], [8, 9], [4, 13]. Номеру 1 соответствуют страницы 2, 6, 10, 14, а номеру 2 — страницы 15, 11, 7, 3. Поэтому на нечётные листы попадут страницы [2, 15], [6, 11], [10, 7], [14, 3]. Если число страниц не кратно числу страниц в блоке, программа pstops добавит пустые страницы.

Небольшое изменение в командной строке позволит разместить по 4 страницы на листе размером A3, вдвое большем A4:

```
pstops.exe -pa5 -d 8:-4U(1w,2h)+3U(2w,2h)+-7(0w,0h)+0(1w,0h), \
    2U(1w,2h)+-5U(2w,2h)+1(0w,0h)+-6(1w,0h) note.ps a3_8.ps
```

При этом две страницы в верхнем ряду на каждом листе будут напечатаны «вверх ногами» (рис. 17.3), на что указывает буква `U` в их спецификации.

Напечатав чётные страницы полученного файла `a3_8.ps` на обороте, сложив каждый лист вчетверо, сложив затем листы в стопку и обрезав по внешним краям, вы получите тетрадь с правильным порядком страниц. В типографиях делают примерно то же, но только на листе бумаги располагают гораздо больше страниц.

Мы не будем комментировать последний в этой главе пример, полагая, что принцип ясен. Добавим только, что страницы можно поворачивать вправо, предваряя их координаты буквой `R`. Можно также изменять масштаб страницы, указывая масштабный множитель перед координатами страницы. Например, чтобы разместить две страницы размера A4 на листе того же размера в естественном порядке, достаточно выполнить следующую команду:

```
pstops.exe -pa4 -d1 2:0L@.7(1w,0h)+1L@.7(1w,0.5h) note.ps a4(2).ps
```

Масштабный множитель `@.7` уменьшает линейные размеры каждой страницы до 70% начальной величины.

Чем проще изделие, тем сложнее
руководство пользователя.
Теорема Чеботаева

Глава 18

Окно в интернет

Бурное развитие интернета и электронного документооборота не в последнюю очередь стало возможным в результате изобретения языка разметки гипертекстовых документов HTML. Существуют более 20 программ для преобразования документов LaTeX в формат HTML (HyperText Markup Language), но такое разнообразие свидетельствует не столько о потребности в таких программах, сколько о нерешённости многих проблем. Не вдаваясь в их обсуждение, мы решили отказаться от нашего первоначального намерения рассказать об одной из программ конвертации документов LaTeX в формат HTML и ограничиться рассмотрением способов преобразования документов LaTeX в формат PDF. Вероятно, время для повсеместного перехода от разметки LaTeX к разметке HTML ещё не пришло или, напротив, уже упущено. Формат PDF (Portable Document Format), появившись позже HTML, де-факто стал мировым стандартом обмена электронными документами, обеспечивая их визуальную аутентичность на любых компьютерах. Собственно, мы уже рассказали о PDF практически всё, что может пригодиться пользователю издательской системы LaTeX. Способы получения документов PDF из исходных текстов с разметкой LaTeX мы обсудили в конце 1 главы. Особенности вставки рисунков в документы PDF освещены в главе 10. В главе 16 показано, что существовавшие когда-то различия в использовании шрифтов в документах DVI и PDF в настоящее время сведены на нет.

В данной главе мы расскажем, как документ LaTeX сделать *гипертекстовым*, добавив возможность перехода к другой части документа щелчком указателя «мышки» по *гиперссылке* в окне обозревателя, будь то обозреватель документов DVI, PDF или HTML. В первом приближении для получения гипертекстового документа достаточно загрузить пакет hyperref:

```
\usepackage{hyperref}
```

Тогда при просмотре откомпилированного документа в окне обозревателя ссылки на литературные источники, разделы документа, уравнения и т. п. приобретут свойства гиперссылок. Обычно гиперссылки выделены цветом или подчёркнуты. При щелчке указателем «мышки» по гиперссылке произойдёт переход в список литературы, к началу раздела, на уравнение с указанным номером и т. д.

18.1. Пакет hyperref

Пакет hyperref Хейко Обердиека (Oberdiek, Heiko) и Себастьяна Ратца (Rahtz, Sebastian) переопределяет множество команд LaTeX'а, вторгаясь в заповедные области. Косвенным свидетельством столь бесцеремонных действий являются многочисленные сообщения об ошибках при первой компиляции документа после подключения пакета или, напротив, после его отключения. Все сообщения нужно игнорировать, если они не появляются вновь при второй или третьей попытке выполнить компиляцию. В списке загружаемых пакетов hyperref, как правило, должен стоять последним, так как он модифицирует работу других пакетов. Пакет вводит новые команды для организации связей как между различными частями одного документа, так и для перехода к другим документам. За рамками нашего рассказа окажутся команды для взаимодействия с элементами меню программы Adobe Reader, предназначенной для просмотра документов PDF. Мы также обойдём молчанием средства создания форм. Формы являются чуть ли не обязательной составной частью HTML страниц на многих веб-сайтах, но даже на сайте компании Adobe трудно найти формы, размещённые в документах PDF. Кроме того, мы не предполагаем, что Читатель хотя бы вкратце знаком с основами языка разметки HTML, без чего рассказ о создании форм совершенно невозможен. Тем не менее мы рассчитываем, что Читатель всё-таки имеет некоторые познания в области Интернет-навигации или по крайней мере осведомлен о том, что адрес любого документа в Интернете должен начинаться с указания протокола, такого как `http://` или `ftp://`.

18.1.1. Опции пакета hyperref

Действиями пакета hyperref можно управлять, задавая определённые параметры (опции) при его загрузке. Параметры перечисляются в необязательном аргументе декларации `\usepackage` через запятую в виде пар *key*=*value*, где *key* — имя параметра, а *value* — его значение. Во многих случаях значение *value* (вместе со знаком =) можно опустить, использовав более привычный способ указания параметров загрузки без значений. Так можно поступить, если значение по умолчанию пусто или равно `true`. Если значение состоит из нескольких слов, его нужно заключить в фигурные скобки, как в следующем примере:

```
\usepackage[
    pdftitle={LaTeX. A document Preparation System},
    pdfauthor={L. Lamport},
    colorlinks=true
]{hyperref}
```

Здесь первые два параметра предназначены для занесения в свойства документа PDF его названия и имени автора. Увы, русские буквы нельзя использовать в необязательном аргументе `\usepackage`, поэтому тем же способом невозможно заполнить свойства документа на русском языке. К счастью, авторы пакета

hyperref предусмотрели ещё один способ передачи пакету параметров через аргумент декларации

`\hypersetup{options}` (hyperref)

Это было сделано прежде всего потому, что список параметров hyperref может быть очень длинным, но для подготовки документов на русском языке такой способ просто незаменим:

```
\usepackage[unicode,colorlinks]{hyperref}
\begin{document}
\hypersetup{
    pdftitle={Пример использования пакета hyperref},
    pdfauthor={Игорь А. Котельников, Платон З. Чеботаев}
}
```

Следует обратить внимание на появление параметра unicode. Только при его наличии русские буквы правильно отображаются в свойствах документа[1] и так называемых закладках (bookmarks).

Выбор драйвера

Результат работы пакета hyperref зависит от предназначения документа, точнее от программы, которую предполагается использовать для просмотра документа в электронном виде. Сообразно этому в списке параметров \usepackage, вообще говоря, следует указать драйвер, аналогично тому, как это делают при использовании графических пакетов (глава 10). Однако во многих случаях пакет без подсказки способен определить, какой компилятор используется для обработки документа, и самостоятельно выбрать правильный драйвер. Например, для компилятора pdflatex автоматически будет выбран драйвер pdftex, а для коммерческого компилятора vtex корпорации MicroPress будет активизирован одноимённый параметр vtex. В иных случаях по умолчанию будет выбран драйвер hypertex. Многие обозреватели документов DVI, в том числе YAP, прекрасно работают с этим драйвером. Но если документ DVI должен быть преобразован в PostScript, следует явно указать параметр dvips. Если затем документ PostScript будет преобразован в PDF, то нужно выбрать драйвер ps2pdf. Если документ DVI предполагается напрямую преобразовать в PDF, следует предпочесть dvipdfm. К счастью, подобные запутанные многоступенчатые способы получения документов PDF на наших глазах становятся достоянием истории. Тем не менее мы перечислим все имеющиеся варианты выбора драйвера:

dvipdf | dvipdfm | dvips | dvipsone | dviwindo | ps2pdf | tex4ht | textures | hypertex | latex2html | pdftex | vtex

[1] Многие пользователи не придают большого значения правильному заполнению свойств документа, не осознавая, что поисковые машины неправильно индексируют подобные документы и, как следствие, не находят их по запросу других пользователей.

Драйверы tex4ht и latex2html используются в связке с одноимёнными программами конвертации документов LaTeX в формат HTML. Для любого параметра, соответствующего имени драйвера, значение можно опустить. Совершенно не обязательно писать pdftex=true, поскольку краткое pdftex означает буквально то же самое.

Режим компиляции

Следующая пара альтернативных параметров дублирует одноимённые опции классов. Она служит для выбора чернового (dtaft) или окончательного (final) варианта компиляции документа:

draft[=false] | final[=true]

Здесь в квадратных скобках после знака равенства дано значение по умолчанию. Простое перечисление final (без значения) в необязательном аргументе декларации \usepackage эквивалентно final=true и имеет смысл, только если документ в целом компилируется с опцией draft в необязательном аргументе \documentclass; режим final и так используется по умолчанию (поскольку значение final по умолчанию равно true, т. е. *истине*). Напротив, параметр draft изменяет режим, используемый по умолчанию, поскольку draft без значения эквивалентен draft=true, тогда как значение draft по умолчание равно false (*ложь*). Параметры draft и final могут принимать только два значения: true и false, как и некоторые другие параметры пакета hyperref, перечисленные ниже. Наличие двух альтернативных параметров, каждый из которых может принимать два альтернативных значения, дважды избыточно. Действительно, final=false — это то же самое, что draft=true. Но такова уж плата за поддержание универсального способа обращения со всеми параметрами пакета hyperref!

Увлёкшись объяснением способа интерпретации необязательных параметров, которые могут принимать булевы значения true или false, мы не сказали главного. В черновом режиме гиперссылки не создаются, а все опции, управляющие созданием гиперссылок, отключены.

Управление гиперссылками

Большая группа параметров управляет размещением гиперссылок в документе. В квадратных скобках приведено значение параметра по умолчанию.

breaklinks[=false] — значение true устанавливает, что часть текста гиперссылки, перенесённая на следующую строку, оформляется в виде отдельной гиперссылки. Значение true выбирается по умолчанию при использовании драйвера pdftex, поскольку в документе PDF текст гиперссылки может размещаться только в одной строке.

pageanchor[=true] — значение true делает возможным переход на любую страницу по её номеру. Пользователи LaTeX'а могут представить себе действие этого параметра как размещение команды \label в левом верхнем углу каждой

страницы. Значение `false` сделает невозможным переход к нужной странице из оглавления, созданного командой `\tableofcontents`.

`plainpages`[`=true`] — значение `true` использует для адресации гиперссылок арабские цифры в номерах страниц, игнорируя форматирование номеров, произведённое классом печатного документа или другими пакетами. Если в исходном тексте используется команда `\pagenumbering` (явно или неявно через декларации `\frontmatter` и `\mainmatter` в классе `book`), опцию `plainpages` следует отключить, так `\pagenumbering` начинает счёт страниц каждый раз заново (в результате две страницы, например 4 и iv, будут иметь одинаковые адреса).

`raiselinks`[`=false`] — при использовании драйвера `hypertex` значение `true` отражает истинную высоту гиперссылки. По умолчанию, когда используется значение `false`, высота текста гиперссылки вычисляется как расстояние между строками (при том, что текст может содержать рисунок).

`backref`[`=false`] — значение `true` вставляет обратные ссылки на номера разделов в конец каждого элемента библиографического указателя. Это позволяет щелчком указателя «мышки» по обратной ссылке переходить из списка литературы в то место документа, где находится ссылка на цитированную литературу. Данный параметр работает при условии, что между отдельными записями в списке литературы имеется пустая строка.

`pagebackref`[`=false`] — значение `true` вставляет обратные ссылки на номера страниц в конце каждого элемента библиографического указателя.

`hyperindex`[`=false`] — значение `true` трансформирует текст элементов алфавитного указателя в гиперссылки.

`hyperfigures`[`=false`] — значение `true` привязывает гиперссылки к плавающим объектам.

`linktocpage`[`=false`] — значение `true` привязывает гиперссылки в оглавлении, списке рисунков и таблиц к номерам страниц, а не тексту.

`extension=`*dvi* — расширение по умолчанию имени файла в гиперссылке, созданной командами `\href` (см. 18.1.2). Для документов PDF расширение по умолчанию есть `pdf`.

Ссылки, как и текст, на который они указывают, могут быть выделены цветом или обведены цветной рамочкой. Выбор одного из двух основных вариантов оформления ссылок определяется значением опции `colorlinks`. Цвета, используемые для выделения, могут быть заданы через значения опций, перечисленных ниже. Цвет выделяемого текста должен быть назван по имени, которое определено ранее согласно правилам, установленным для пакета color (раздел 10.7). Цвет рамки вокруг ссылки можно задать только тремя числами, соответствующими яркости красного, зеленого и синего оттенков, причём каждое число может принимать значение от 0 до 1. В приводимом ниже списке значения по умолчанию указаны после знака равенства рядом с именем опции.

`colorlinks`[`=false`] — значение `true` включает схему выделения ссылок цветом. Значение `false`, используемое по умолчанию (кроме случая, когда выбран драйвер `tex4ht` или `dviwindo`), соответствует выделению ссылок цветными рамками в документах PDF. В документах других форматов ссылки будут выделены подчёркиванием. Цвет зависит от типа ссылки. Принято выделять разными цветами ссылки на цитированную литературу, ссылки на страницы, URL ресурсов в интернете, ссылки на локальные файлы.

`anchorcolor`[`=black`] — цвет текста, на который имеется гиперссылка.

`citecolor`[`=green`] | `citebordercolor`[`={0 1 0}`] — цвет ссылки на цитированную литературу в библиографическом указателе или цвет рамки вокруг этой ссылки. Значения `citecolor` и `citebordercolor` используются соответственно при `colorlinks=true` и `colorlinks=false`.

`filecolor`[`=magenta`] | `filebordercolor`[`={0 .5 .5}`] — цвет ссылки на файл или цвет рамки вокруг неё.

`urlcolor`[`=cyan`] | `urlbordercolor`[`={0 1 1}`] — цвет ссылки на URL ресурса или цвет рамки вокруг этой ссылки.

`linkcolor`[`=red`] | `linkbordercolor`[`={1 0 0}`] — цвет ссылки на иные объекты (например, на номер раздела или номер уравнения) или цвет рамки вокруг этой ссылки.

`pagecolor`[`=red`] | `pagebordercolor`[`={1 1 0}`] — цвет ссылки на страницу или цвет рамки вокруг этой ссылки.

`pdfborder`[`={0 0 1}`] — стиль рамки вокруг ссылки. В настоящее время первые два числа игнорируются, но тем не менее должны присутствовать. Третье число задаёт толщину рамки вокруг ссылки в документе PDF. По умолчанию толщина равна `1pt`, если не использована опция `colorlinks`; в последнем случае толщина рамки по умолчанию равна `0pt`.

Управление закладками

Следующая группа опций предназначена для настройки закладок (bookmarks) в документе PDF. В документах других форматов эти опции просто игнорируются. Закладки обычно отображаются в левой части окна программы Adobe Reader. Они используются для быстрой навигации в пределах одного документа и чаще всего дублируют оглавление.

`bookmarks`[`=true`] — значение `true` формирует закладки из оглавления. Требуется, как минимум, двукратная компиляция исходного текста. Если в оглавлении имеются команды LaTeX'а, может потребоваться дополнительное редактирование файла закладок (он имеет расширение `out`). Чтобы предотвратить замещение файла закладок при последующих компиляциях, нужно добавить в него строку `\let\WriteBookmarks\relax`.

bookmarksopen[=false] — значение true означает, что дерево закладок должно быть раскрыто при открытии документа.

bookmarksnumbered[=false] — значение true означает, что закладки будут включать номера разделов.

bookmarksopenlevel — уровень, до которого раскрыты закладки при открытии файла. По умолчанию раскрыты все закладки, если bookmarksopen=true.

bookmarkstype[=toc] — указывает, какой файл содержит информацию для закладок.

Свойства документа PDF

Ряд параметров используется для описания свойств документа PDF. Они игнорируются, если в результате компиляции будет создан документ формата, отличного от PDF.

pdftitle[={}] — заносит название документа в поле title свойств файла pdf.

pdfauthor[={}] — заносит имена авторов в поле author свойств файла.

pdfsubject[={}] — заносит тему документа в поле subject свойств файла.

pdfkeywords[={}] — заносит ключевые слова в поле keywords свойств файла.

pdfcreator[={LaTeX with hyperref package}] — заносит сведения о программе, использованной для создания документа, в поле creator свойств файла.

pdfproducer[={pdfTeX-1.10b}] — заносит сведения о программе, использованной для создания файла, в поле producer свойств файла. Значение по умолчанию зависит от типа и версии компилятора.

Управление окном просмотра документа PDF

Ещё одна группа параметров позволяет настроить начальный вид документа PDF при его открытии.

pdfpagemode[=None] — определяет вид окна программы Adobe Reader при открытии документа. Возможные значения: None (показать только документ), UseThumbs (показывать иконки страниц), UseOutlines (показывать закладки) и FullScreen (показать документ во весь экран). По умолчанию используется значение UseOutlines, если выбрана опция bookmarks, и None в ином случае.

pdfstartpage[=1] — определяет номер страницы, которая будет показана при открытии документа.

pdfstartview[=FitH] — определяет видимую часть страницы, которая будет показана при открытии документа. Значение по умолчанию FitH соответствует такому масштабу, который обеспечивает подгонку ширины страницы по размеру окна программы Adobe Reader. Другие возможные значения перечислены ниже.

`XYZ` *left top zoom* — выбрать заданный масштаб. Размер страницы изменяется в *zoom* раз, после чего страница прокручивается так, чтобы её горизонтальные и вертикальные координаты *left* и *top* находились в левом верхнем углу окна. Если страница после изменения размера целиком помещается в окне, прокручивание не производится. Значение `null` для любого параметра *left*, *top* или *zoom* поддерживает неизменным текущее значение этого параметра. Для *zoom* значение `0` эквивалентно `null`. Значения координат задаются в больших пунктах (72 bp равны 1 дюйму). Для страницы формата A4 левый нижний угол имеет координаты (0, 0), а правый верхний — (595, 842).

`Fit` — подогнать масштаб по ширине страницы. Страница целиком подгоняется под размер окна и центрируется.

`FitH` *top* — подогнать масштаб по ширине страницы. Страница подгоняется по ширине окна и, если указан параметр *top*, прокручивается по вертикали так, чтобы её вертикальная координата *top* совпадала с верхней границей окна. Если страница после подгонки по ширине окна целиком помещается в нём, прокручивание не производится.

`FitV` *left* — подогнать масштаб по высоте страницы. Страница подгоняется по высоте окна и, если указан параметр *left*, прокручивается по горизонтали так, чтобы её координата *left* совпадала с левой границей окна. Если страница после подгонки по высоте окна целиком помещается в нём, прокручивание не производится.

`FitB` — подогнать масштаб по размеру текста на странице. Прямоугольник, ограничивающий часть страницы, занятую текстом и рисунками, подгоняется под размер окна и центрируется.

`FitBH` *top* — подогнать масштаб по ширине текста на странице. Ширина ограничивающего прямоугольника подгоняется по ширине окна и, если указан параметр *top*, страница прокручивается по вертикали так, чтобы её координата *top* совпадала с верхней границей окна. Если страница после изменения размера целиком помещается в окне, прокручивание не производится.

`FitBV` *left* — подогнать масштаб по высоте текста на странице. Высота ограничивающего прямоугольника подгоняется по высоте окна и, если указан параметр *left*, страница прокручивается по горизонтали так, чтобы её координата *left* совпадала с левым краем окна. Если страница после изменения размера целиком помещается в окне, прокручивание не производится.

`FitR` *left bottom right top* — подогнать масштаб по заданным размерам. Прямоугольная область страницы, заданная координатами *left*, *bottom*, *right* и *top*, целиком подгоняется под размер окна и центрируется.

`pdfview` — определяет видимую часть страницы, которая будет показана при переходе на неё по гиперссылке. Значение по умолчанию совпадает со значением параметра `pdfstartview`.

`pdfpagescrop` — устанавливает размеры страницы подобно BoundingBox в файлах EPS. Следует указать 4 числа *left*, *bottom*, *right* и *top*, последовательно

задав абсциссу и ординату левого нижнего и правого верхнего углов (в единицах bp) прямоугольника, ограничивающего видимую область страницы.

`pdfnewwindow`[`=false`] — значение `true` открывает ссылку на другой документ в новом окне.

`pdfcenterwindow`[`=false`] — значение `true` размещает окно документа в центре экрана.

`pdffitwindow`[`=false`] — значение `true` подгоняет размер окна под размер документа при его открытии.

`pdfmenubar`[`=true`] — значение `true` показывает меню.

`pdftoolbar`[`=true`] — значение `true` показывает панель управления.

`pdfwindowui`[`=true`] — значение `true` показывает пользовательские элементы управления, включая левую панель и строку состояния внизу окна.

Прочие опции

В последнюю группу опций попадают все параметры, которые сложно причислить к любой из рассмотренных ранее групп.

`debug`[`=false`], `verbose`[`=false`] — две одинаковые по действию опции расширяют объём информации, записываемый в файл протокола при компиляции документа.

`baseurl`=*baseurl* — устанавливает базовую часть *baseurl* адресов гиперссылок URL (Universal Resource Locator). Базовая часть добавляется в начало адреса любой гиперссылки, заданной без указания протокола (см. ниже).

`unicode`[`=false`] — форсирует переход на кодировку Unicode.

18.1.2. Пользовательские команды

Используйте команду

`\href{`*url*`}{`*text*`}` (hyperref)

чтобы создать гиперссылку на адрес *url* (Universal Resource Locator) документа или ресурса в сети интернет. В самом общем случае *url* имеет вид

protocol://*domain*:*port*/*path*/*file-name*#*hash*?*search*

где *protocol* — протокол доступа к ресурсу, *domain* — домен, где размещён ресурс, *port* — порт, через который производится доступ, /*path* — путь к ресурсу внутри домена, /*file-name* — имя файла, #*hash* — закладка внутри ресурса, ?*search* — строка поиска. Любая часть *url* может отсутствовать.

Примерами протоколов являются `html://` (обеспечивает доступ к Web сайтам), `ftp://` (обращение к серверам ftp), `file://` (обращение к ресурсам в пределах локальной файловой системы), `res://` (обращение к ресурсам операционной

системы), `mms://` (потоковое мультимедиа-вещание). Доменное имя, например `www.ctan.org`, ставится в соответствие уникальному адресу компьютера в мировой сети интернет. Порт чаще всего выбирается по умолчанию соответственно указанному протоколу. Например, протоколу `http://` по умолчанию сопоставлен порт `:80`. Путь на ресурс в пределах компьютера также зависит от выбранного протокола. Для протокола `file://` — это просто полный или относительный путь до каталога, где размещён нужный файл. Строка поиска используется при обращении к динамическим ресурсам, которые способны возвращать разный результат в зависимости от значениий параметров, переданных в этой строке. Использование закладок поясняет следующий пример:

```
Вот пример на
\href{file://primer#page.2}{странице 2}.
```

Вот пример на странице 2.

Если щёлкнуть на подчёркнутом тексте в окне обозревателя, будет выполнен переход на вторую страницу в файле `primer.dvi`, если текущий документ просматривается в обозревателе документов DVI, или на вторую страницу в файле `primer.pdf`, если просматривается документ PDF. Предполагается, что оба файла находятся в том же каталоге, что и документ, который содержит гиперссылку. Чтобы исключить неопределённость с расширением имени файла, можно либо явно указать в гиперссылке имя файла вместе с расширением, либо задать расширение по умолчанию с помощью параметра `extension` при загрузке пакета hyperref. Как ясно из примера, команда `\href` использует свой первый аргумент *url* в качестве адреса перехода, а второй аргумент *text* выводит в текст документа. Вместо относительного пути на каталог, где расположен файл, можно указать путь полностью:

```
Вот пример на
\href{file://e:/platon/primer#page.2}
{странице 2}.
```

Вот пример на странице 2.

Если большинство адресов имеет общую часть, её можно объявить с помощью декларации

`\hyperbaseurl{`*url*`}` (hyperref)

или через параметр `baseurl`. Знатоки языка разметки HTML легко догадаются, что `\hyperbaseurl` имеет то же действие, что и таг `<baseurl>` в документах HTML. Объявленную общую часть *url* можно исключить из адресов любых гиперссылок. Если протокол отсутствует в *url* (даже после присоединения общей части), обозреватель подбирает протокол по своему разумению.

Команда

`\hypertarget{`*key*`}{`*label-text*`}` (hyperref)

присваивает фрагменту текста *label-text* закладку с меткой вида *key*. На эту закладку затем можно дать гиперссылку различными способами. Можно использовать команду `\href`, указав в качестве *url* строку `#`*key* (если переход производится в пределах одного документа). Можно также использовать команду

`\hyperlink{`*key*`}{`*text*`}` (hyperref)

Добавить гиперссылку на метку, поставленную стандартной командой LaTeX'а `\label{`*key*`}`, проще всего при помощи команды `\hyperlink`, указав в первом аргументе тот же ключ *key*. Стандартная команда `\ref{`*key*`}` делает примерно то же, но в качестве текста *text* обычно печатает номер объекта, на который дана ссылка.

Мы не упомянули ещё несколько команд, которые пакет hyperref вводит для организации гиперссылок. И поэтому, завершая книгу, вновь повторим слова, сказанные нами в её начале.

> *Из нескольких решений одной задачи мы стремились выбрать одно, самое эффективное, не затрудняя Читателя сравнением всех альтернатив. Сделать правильный выбор часто составляет самую сложную часть задачи. Именно эту часть мы постарались решить, отбирая самое необходимое, чтобы очистить LaTeX от археологических наслоений.*

Остаётся просто опыт, который зовётся
случайным, если приходит сам,
и экспериментом, если его отыскивают.
Ф. Бэкон. Новый Органон

Приложение A

Режим эмуляции LaTeX 2.09

LaTeX 2ε может компилировать почти любой входной файл, предназначенный для LaTeX версии 2.09, за счёт перехода в *режим эмуляции* LaTeX 2.09. Оговорку приходится делать из-за наличия некоторого количества пакетов, которые использовали недокументированные команды LaTeX 2.09 низкого уровня. Большинство старых пакетов успешно работают с LaTeX 2ε, а лучший способ узнать, пригоден тот или иной пакет,— просто испытать его.

Режим эмуляции есть почти полная имитация LaTeX 2.09 как на внешнем (пользовательском), так и на внутреннем уровнях. Однако такая имитация достигается ценой уменьшения скорости компиляции примерно на 50% по сравнению с компиляцией с настоящим форматом LaTeX 2.09. Большинство нововведений, появившихся в LaTeX 2ε, отключено в режиме эмуляции LaTeX 2.09.

LaTeX 2ε автоматически переходит в режим эмуляции, когда обнаруживает декларацию

```
\documentstyle[options,pkgs]{class}
```

вместо `\documentclass`. Режим эмуляции предназначен для компиляции старых файлов. Его не следует использовать для компиляции вновь подготавливаемых документов. Чтобы компилировать старые файлы быстрее и получить доступ к новым возможностям, открываемым переходом к LaTeX 2ε, следует преобразовать `\documentstyle` в две декларации

```
\documentclass[options]{class}
\usepackage{latexsym,pkgs}
```

Чтобы определить, какие опции из `\documentstyle` следует перенести в список опций `\documentclass`, а какие перенести в `\usepackage`, следует изучить список существующих опций декларации `\documentclass`, приведённый в разделе 3.2. Всё, что отсутствует в этом списке, является пакетом, а пакеты загружаются при помощи `\usepackage`. Например,

```
\documentstyle[12pt,russian]{article}
```

надо переделать в

```
\documentclass[12pt]{article}
\usepackage{latexsym}
\usepackage[cp866]{inputenc}
\usepackage[russian]{babel}
```

Пакет latexsym содержит определения нескольких редко используемых команд, которые имелись в формате LaTeX 2.09, но удалены из LaTeX 2_ε. Эти команды выделены в таблицах 6.4, 6.5, 6.6 и 6.9. Опция `russian` в LaTeX 2.09 относилась к числу недокументированных; в разных диалектах LaTeX 2.09 её действие было различно.

Совершенно новым моментом для пользователей, знакомых с LaTeX 2.09, является необходимость указывать кодировку исходного текста во входном файле. Кодировка указывается в необязательном аргументе декларации `\usepackage`, которая загружает пакет inputenc. В данном примере использована опция `cp866`, соответствующая кодировке русской версии операционной системы MS DOS. Если исходный текст документа подготовлен в операционной системе Unix, вероятно, следует выбрать опцию `koi8-r`, `koi8-ru` или `koi8-u`.

В LaTeX 2_ε поддержку русского языка в числе десятков других языков народов мира осуществляет пакет babel. В документах на русском языке опция `russian` должна быть указана в необязательном аргументе декларации `\usepackage`, которая загружает пакет babel. Не будет ошибкой, если опции пакетов перенести в необязательный аргумент декларации `\documentclass`:

```
\documentclass[12pt,cp866,russian]{article}
\usepackage{latexsym,inputenc,babel}
```

Однако расстановка опций «по месту употребления пакета» авторам данной книги представляется более правильным решением, поскольку при удалении пакета нужно удалять и принадлежащие ему опции из `\documentclass`.

Большинство старых документов LaTeX 2.09 успешно компилируются в естественном режиме LaTeX 2_ε после внесения указанных выше изменений. Однако некоторые документы, успешно компилируемые в режиме эмуляции LaTeX 2.09, не удаётся без ошибок обработать в естественном режиме LaTeX 2_ε. Некоторые документы вызывают появление сообщений об ошибках из-за улучшенной диагностики ошибок, введённой в LaTeX 2_ε. Некоторые старые пакеты работают с LaTeX 2_ε только в режиме эмуляции. И тут мы можем повторить, что лучший способ выяснить, будут ли проблемы с компиляцией в естественном режиме,— просто попробовать её выполнить!

Затруднения возникают чаще всего из-за команд переключения шрифтов. Эти команды либо вызывают сообщения об ошибках, либо делают не то, что от них ожидалось.

В первом случае возможно, что входной файл или стиль (в LaTeX 2_ε стиль называется пакетом), использованный в нём, содержит устаревшие внутренние команды, которые более не поддерживаются. Вот их неполный перечень:

```
\tenrm \elvrm \twlrm ...
\tenbf \elvbf \twlbf ...
\tensf \elvsf \twlsf ...
⋮
```

Их использование вызывает сообщение об ошибке типа

```
! Undefined control sequence.
l.5 \tenrm
```

Лучше всего получить более свежую версию пакета, не содержащую такие команды. Если это невозможно, то в качестве последнего средства следует загрузить пакет rawfonts из коллекции tools, который обеспечивает доступ более чем к 70 шрифтам через устаревшие команды. Можно ограничить количество загружаемых шрифтов, используя опцию `only`, перечислив вслед за ней только те шрифты, которые нужны. Например,

```
\usepackage[only,tenrm,tenbf]{rawfonts}
```

загружает только шрифты, вызываемые командами `\tenrm` и `\tenbf`.

Если после приведения документа к стандарту LaTeX 2ε сообщения об ошибках не генерируются, но команды переключения шрифтов делают не то, что ожидалось (особенно в математических формулах), лучше всего их заменить на команды, описанные в разделе 1.11. Другой вариант выхода из встретившихся затруднений — загрузить пакет oldlfont.

Существовал также диалект формата LaTeX 2.09, в котором двухбуквенные команды типа `\sf` и `\it` переключали только один атрибут шрифта, т. е. действовали так же, как в LaTeX 2ε работают декларации `\sffamily` и `\itshape`. Если есть потребность восстановить такое поведение команд `\sf`, `\it` и им подобных, нужно загрузить пакет newlfont.

Одинаковые ошибки необязательно делать
каждый раз, достаточно сделать одну,
а затем обращаться к ней по мере
необходимости.
В. Тихонов. Теория ошибок

Приложение B

Ошибки

В разделе 1.13 изложены первоначальные сведения об ошибках и способах их локализации во входном файле. Данное приложение содержит перечень возможных сообщений об ошибках. Строго говоря, этот перечень не охватывает все возможные кризисные ситуации, поскольку единственная ошибка может запутать компилятор настолько, что он выдаст множество сообщений об ошибках, которых на самом деле нет. Отсюда следует правило № 1: исправлять ошибки следует с самой первой. Разумеется, всякое правило имеет исключения, но каждое исключение только подтверждает правило. Когда компилятор записывает в выходной файл очередную готовую страницу текста, он обычно выходит из нокдауна, вызванного предыдущими ошибками, так что следующее диагностическое сообщение скорее всего указывает на ошибку, действительно существующую.

В особо сложных случаях компилятор может не распознать причину ошибки. Мы не рассматриваем такие «мистические» ошибки детально, ограничиваясь лишь общими, но эффективными рекомендациями по их устранению. Мы также не перечисляем сообщения об ошибках, которые могут генерировать многочисленные пакеты, описанные в нашей книге. Наиболее типичные кризисные ситуации, связанные с работой пакетов, рассмотрены в соответствующих главах книги.

B.1. Установка таблицы переносов

Одной из проблем, с которой сталкивается пользователь, приступивший к работе с системой LaTeX, является видимое отсутствие переносов в текстах на русском языке. Хотя поддержка русского языка с 1999 года является обязательной составной частью любой версии LaTeX, по умолчанию ни русские шрифты, ни алгоритм переноса русских слов не устанавливаются. После завершения стандартной процедуры установки необходимо предпринять ещё ряд шагов, чтобы иметь возможность работать с русскими текстами.

1. Если в откомпилированном документе полностью отсутствуют переносы в русских словах, то прежде всего следует проверить, была ли при установке системы LaTeX подключена таблица переносов русского языка. В начале компиляции любого документа LaTeX выводит на экран монитора и записывает в файл

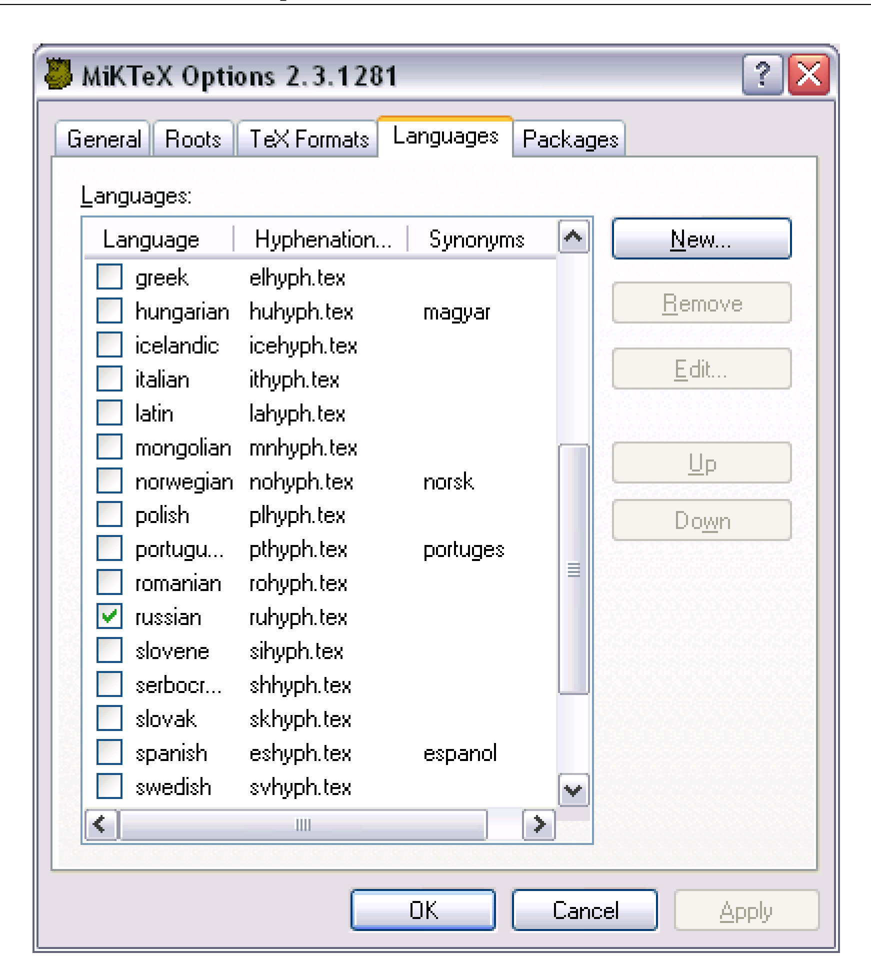


Рис. B.1. Выбор таблицы переносов

протокола список доступных таблиц переносов:

```
Babel <v3.7h> and hyphenation patterns for american, french,
german, ngerman, russian, nohyphenation, loaded.
```

Если слово `russian` присутствует, как в данном примере, с таблицей переносов всё в порядке. Если же его нет, нужно перегенерировать формат LaTeX, предварительно добавив русский язык к списку поддерживаемых языков. На рисунке B.1 показана вкладка **Languages** мастера настройки **MiKTeX Options** с отмеченным русским языком. Аналогичные средства настройки имеются в других реализациях системы LaTeX. Для полной уверенности нужно проверить, что файл `ruhyph.tex` с таблицей переносов русского языка физически присутствует на жёстком диске компьютера. Это можно сделать непосредственно с вкладки

Languages, изображённой на рисунке B.1, если нажать кнопку **Edit**. Если файл отсутствует, его надо установить с помощью мастера установки пакетов **MiKTeX Package Manager**. Затем необходимо перегенерировать формат LaTeX. В мастере управления **MiKTeX Options** для этого можно нажать кнопку **Update Now** на вкладке **General**.

2. Если с таблицей переносов всё в порядке, а русский текст правильно читается в откомпилированном документе, то алгоритм переносов должен работать и чётко указать причину, почему он мог бы не действовать, заочно не представляется возможным. Можно порекомендовать проверить наличие декларации

```
\usepackage[russian]{babel}
```

в преамбуле входного файла.

3. Если кодировка исходного текста указана неправильно, это может привести к неправильной расстановке переносов. Кодировка исходного текста указывается в необязательном аргументе пакета inputenc. Однако неправильная кодировка имеет более заметный визуальный эффект: текст становится бессмысленным. Для текста, подготовленного в кодировке русских версий Windows, нужно загружать пакет inputenc с опцией `cp1251`:

```
\usepackage[cp1251]{inputenc}
```

Внутренняя кодировка русских шрифтов, которые загружает пакет babel с опцией `russian`, очень близка к кодовой странице `cp1251`. Поэтому текст в печатном документе будет читаться почти правильно, если пакет inputenc вообще не загружен. Отличие можно зафиксировать по букве ё — она будет отображена другим символом.

4. Если русские буквы не отображаются вообще или заменены латинскими, в системе отсутствуют русские шрифты для LaTeX'а. В таком случае необходимо прежде всего убедиться, что установлены кириллические шрифты METAFONT семейства LH. Если установлены и шрифты LH, и шрифты CM-Super, пропуски русских букв в откомпилированном документе обычно означают, что шрифты CM-Super установлены неправильно. В этом случае следует обратиться к документации, сопровождающей эти шрифты.

5. Видимая деградация качества русских шрифтов в документах формата PDF или PostScript также означает, что шрифты CM-Super не установлены или установлены неправильно.

B.2. Как искать ошибки

В сообщениях об ошибках указывается номер строки, где обнаружена ошибка. Например, `l.50` в строке индикации ошибки (раздел 1.13) означает, что ошибка обнаружена, когда обрабатывалась пятидесятая строка от начала файла. Если весь исходный текст записан в одном входном файле, тогда информация о номере

строки однозначно указывает место, где, как полагает компилятор, находится источник проблем. Однако если исходный текст разбит на несколько файлов с помощью команд, описанных в разделе 3.8, необходимо ещё установить, в каком файле обнаружена ошибка. Для этого компилятор выводит на экран (и записывает в файл протокола с расширением `log`) имена обрабатываемых файлов. Всякий раз, когда начинается обработка нового файла, на экран выводится левая круглая скобка «`(`», за которой следует имя файла. Дойдя до конца файла или встретив команду `\endinput`, компилятор выводит «`)`». Предположим, что компилятор вывел следующее сообщение:

```
(C:\My Documents\manual.tex [1] [2] [3] (C:\My Documents\ch1.tex [4]
[5]) [6]
! LaTeX Error: Environment pictre undefined.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.321 \begin{pictre}
                    (100,100)
?
```

Первой левой круглой скобкой компилятор извещает, что начал обработку файла `manual.tex` в каталоге `My Documents` на диске `C`. После вывода первых трёх страниц, номера которых заключены в квадратные скобки, он перешёл к обработке файла `ch1.tex`. Такой переход случается, если корневой файл `manual.tex` содержит команду `\input{ch1}` или `\include{ch1}` (раздел 3.8). После вывода ещё двух страниц с номерами 4 и 5 компилятор успешно закончил обработку этого файла и вернулся к файлу `manual.tex`. А вот после вывода ещё одной страницы с номером 6 он обнаружил ошибку в строке 321. Она находится в файле `manual.tex`. Вслед за номером ошибочной строки компилятор печатает на экране (и записывает в файл протокола) её содержимое в виде двух строк, причём первая строка заканчивается ошибочной командой. В нашем примере строка с ошибкой во входном файле выглядит как

```
\begin{pictre}(100,100)
```

Обнаружив ошибку, компилятор печатает в конце сообщения знак вопроса и приостанавливает работу, ожидая реакции пользователя. В этот момент можно проигнорировать ошибку, нажав клавишу ⟨Enter⟩, чтобы попробовать обнаружить другие ошибки. Однако одна действительная ошибка способна вызвать множество других ошибок, так как одна команда LaTeX'а состоит из множества команд TeX'а. Если ошибок много, можно при очередной остановке ввести с клавиатуры ⟨q⟩⟨Enter⟩, чтобы затем изучить все сообщения об ошибках (они будут записаны в файл протокола, но не будут выводиться на экран). Можно также прекратить компиляцию, введя ⟨x⟩⟨Enter⟩. В нашем примере внимательное изучение сообщения о первой ошибке в строке 321 позволяет легко обнаружить причину ошибки:

вместо несуществующей процедуры `pictre` должно быть `picture`. Такую ошибку легко поправить, нажав клавиши ⟨i⟩⟨Enter⟩, и на последующее приглашение

```
\insert>
```

ввести исправленную команду

```
\insert>\begin{picture}
```

Это позволит избежать последующих наведённых ошибок, но затем всё равно необходимо исправить ошибку во входном файле с помощью текстового редактора. Если ошибку не удаётся легко локализовать в исходном тексте документа, бывает полезно просмотреть результат компиляции в окне браузера.

Все ошибки можно разделить на две группы: ошибки LaTeX'а и ошибки TeX'а. В приведённом выше примере зафиксирована ошибка LaTeX'а, на что указывают слова «`LaTeX error`» в сообщении об ошибке. Далее в двух строках следует рекомендация ещё раз почитать данную книгу или нажать клавиши ⟨H⟩⟨Enter⟩, чтобы получить на экране краткую справку об ошибке (но на английском языке). Затем следует строка индикации ошибки. Она начинается с восклицательного знака и содержит текстовый признак ошибки. По этому признаку следует отыскивать описание ошибки в перечне, приведённом в следующем разделе.

Изменим строку `321` в нашем примере так, чтобы создать ошибку TeX'а:

```
\began{picture}(100,100)
```

В этом случае сообщение об ошибке не будет содержать упоминания о LaTeX'е:

```
! Undefined control sequence.
l.321 \began
            {picture}(100,100)
?
```

Наконец, в особо тяжёлых случаях компилятор выводит звёздочку

```
*
```

и останавливается без какого-либо сообщения. Такое случается, если пропущена команда `\end{document}`, имеется ошибка в процедурах `picture`, `figure` или вообще что-то неладное происходит в файлах, которые компилятор читает или в которые что-то записывает. Если такая ошибка случилась, то можно попытаться ввести `\stop`⟨Enter⟩ без предварительного ⟨i⟩⟨Enter⟩ или же абортировать программу стандартным для используемого компьютера способом (часто помогает одновременное нажатие клавиш ⟨Ctrl⟩ и ⟨c⟩). Поиск ошибки, вызвавшей `*`, бывает нелёгким делом. Как крайнюю меру можно рекомендовать многократные попытки обработать «больной» файл по частям, постепенно отсекая его «здоровые» фрагменты. С этой целью можно перемещать команду `\end{document}` с конца входного файла в подозрительную область. Часть текста во входном файле после команды `\end{document}` игнорируется компилятором, как если бы она действительно была удалена.

Если поиск причины ошибки производит TEXперт (то есть знаток TEX'а), можно заставить LaTeX выводить более подробную информацию об ошибке. Для этого необходимо назначить счётчику

`errorcontextlines`

какое-нибудь значение, большее чем -1, принятое по умолчанию. Например:

```
\setcounter{errorcontextlines}{99}
```

Можно также отключить реальную компиляцию и перевести LaTeX в режим проверки исходного текста, загрузив пакет syntonly. В этом случае LaTeX не создаёт dvi-файл и поэтому работает несколько быстрее.

Иногда ошибка фиксируется при выполнении команды `\begin{document}` или `\end{document}` в момент, когда считываются вспомогательные файлы с расширением `aux`. О том, что имеет место именно такая ситуация, можно судить по появлению их имен на экране монитора непосредственно перед сообщением об ошибке. Ошибка при обработке `\begin{document}` бывает вызвана ошибкой при предыдущей компиляции документа. Причиной ошибки при обработке `\end {document}` чаще всего бывает хрупкая команда (раздел 2.7).

Помимо сообщений об ошибках, компилятор выдаёт предупреждения, фиксируя менее серьёзные проблемы. Предупреждения выводятся на экран и записываются в файл протокола, но обработка входного файла не приостанавливается. Например, сообщение

```
Overfull \hbox (22.53354pt too wide) in paragraph at lines
176--191 []\T2A/cmr/m/n/10 Первой ле-вой круг-лой скоб-кой
\OT1/cmr/m/n/10 L[]T[]X \T2A/ cmr/m/n/10 из-ве-ща-ет, что на-чал
об-ра-бот-ку фай-ла \T2A/cmtt/m/n/10 manual. tex
 []
```

предупреждает, что при форматировании строк 176–191 компилятор не нашёл подходящего места для переноса и одна из строк на 22,53354 pt вышла за правую границу. Предупреждения компилятора чаще всего начинаются со слов «`LaTeX Warning.`» Наряду с ошибками они описаны в разделе B.3.

Причинами наиболее часто встречающихся ошибок могут быть:

- ошибки в имени команды или процедуры, а также несоответствие открывающих и закрывающих фигурных или командных скобок;
- пропуск одного из разделителей математической моды: `\[`, `\]`, `\(`, `\)`, `$`, `$$`;
- пропуск аргумента команды;
- пропуск `[` для выделения необязательного аргумента команды;
- использование команд в неправильной моде, например применение математических команд `^`, `_`, `\alpha` и т. д. в обычном тексте.

В.3. Диагностические сообщения компилятора

В данном разделе в алфавитном порядке перечислены основные диагностические сообщения компилятора. Сообщения об ошибках помечены значком E, а предупреждения компилятора — значком i. Каждое сообщение сопровождается кратким пояснением о возможных причинах его появления и способах устранения возникших проблем.

Сообщения об ошибках обычно начинаются со слов `LaTeX Error`. Если ошибка зафиксирована в командах уровня более низкого, нежели пользовательский, сообщение начинается со знака восклицания `!`.

Предупреждения обычно начинаются со слов `LaTeX Warning`. Если предупреждение выдано системой загрузки шрифтов NFSS (New Font Selection System), оно может начинаться со слов `Font Info`. В отличие от ошибок, предупреждения не приостанавливают обработку входного файла, а знак `?` на экране и в файле протокола компиляции не печатается. Некоторые предупреждения в обобщенном виде дублируются в конце протокола компиляции.

E `\< in mid line.`
(`\<` в середине строки.)
В теле процедуры `tabbing` использована команда `\<` в середине строки. Эта команда может появляться только в начале строки.

E `\begin{`*env1*`} ended by \end{`*env2*`}.`
(`\begin{`*env1*`}` заканчивается `\end{`*env2*`}`.)
Нарушено соответствие командных скобок. Возможно, пропущено имя процедуры (пропущен аргумент команды `\end`) или оно не соответствует имени процедуры в командной скобке `\begin`.

E `\include cannot be nested.`
(Команда `\include` не может быть вложена.)
Команда `\include` использована в файле, который включен в корневой входной файл при помощи другой команды `\include`.

E `\pushtabs and \poptabs don't match.`
(`\pushtabs` не соответствует `\poptabs`.)
Обнаружено несоответствие числа команд `\poptabs` и `\pushtabs` в процедуре `tabbing`.

E `Bad \line or \vector argument.`
(Плохой аргумент команды `\line` или `\vector`.)
Первый аргумент команды `\line` или `\vector`, определяющий наклон линии или вектора, задан неправильно. Ограничения на этот параметр описаны в разделах 9.6.2 и 9.6.3.

E `Bad math environment delimiter.`
(Плохой разделитель математической моды.)
Нарушен баланс скобок, открывающих и закрывающих математическую моду.

Возможно, что скобке \] или \) не предшествует скобка \[или \(. Возможно, что LaTeX встретил скобку \[или \(, когда он уже находился в математической моде.

E `Bad use of \\.`
(Плохое использование команды \\.)
Команда \\ использована между абзацами, где её появление бессмысленно. Возможно также, что она неправильно использована в процедурах позиционирования текста (`center`, `flushright` или `flushleft`).

E `Can be used only in preamble.`
(Может использоваться только в преамбуле.)
После `\begin{document}` найдена декларация, которая может располагаться только в преамбуле (например, `\documentclass`, `\makeindex` или `\usepackage`). Ошибка может быть вызвана также повторной командой `\begin{document}`.

i `Citation` *citation* `on page` *page* `undefined.`
(Ссылка на запись в списке литературы *citation* на странице *page* не определена.)
Метка ссылки в команде `\cite` не была определена командой `\bibitem`.

E `Command name` *name* `already used.`
(Команда с именем *name* уже использована.)
Использована одна из деклараций типа `\new`*cmd*, чтобы определить команду, процедуру или длину с уже существующим именем. Нужно использовать другое имя или переопределить существующую команду или процедуру с помощью `\renew`*cmd*.

i `Command name` *name* `invalid in math mode.`
(Команда с именем *name* не действует в математической моде.)
Это предупреждение может быть также сообщением об ошибке. Оно означает, что имеет место попытка использовать в математическом режиме команду, которая предназначена для применения только в тексте.

E `Command` *name* `not provided in base NFSS.`
(Команда *name* не реализована в базовой схеме NFSS.)
NFSS (New Font Selection Scheme, Новая схема выбора шрифтов) исключила ряд символов, имевшихся в LaTeX 2.09, из формата LaTeX 2_ε. Ошибка генерируется при использовании одной из команд

```
\mho      \Join     \Box \Diamond \leadsto
\sqsubset \sqsupset \lhd \unlhd   \rhd      \unrhd
```

которые теперь определены в пакете latexsym. Добавьте в преамбулу входного файла декларацию `\usepackage{latexsym}`.

E `Command name` *name* `not defined as a math alphabet.`
(Команда *name* не определена в качестве математического алфавита.)
Сообщение вызвано ошибкой в шрифтовом пакете. Обратитесь к автору пакета.

E `Corrupted NFSS tables.`
(Испорчены таблицы NFSS.)
Испорчена система загрузки шрифтов. Например, удалён какой-нибудь файл определения шрифтов (с расширением `fd`). Переустановите LaTeX или пакет, вызвавший ошибку.

E `Counter too large.`
(Слишком большое значение счётчика.)
Значение счётчика, печатаемого буквами, больше числа букв в алфавите текущего языка. Возможно, используется очень длинный список или допущена ошибка в определении счётчика.

E `Double subscript.`
(Двойной нижний индекс.)
Например, выражение `x_{2}_{3}` не имеет смысла. Чтобы получить x_{2_3}, надо набрать `x_{2_{3}}`.

E `Double superscript.`
(Двойной верхний индекс.)
Например, выражение `x^{2}^{3}` не имеет смысла. Чтобы получить x^{2^3}, надо набрать `x^{2^{3}}`.

i `Encoding` ENC1 `has changed to` ENC2 `for...`
(Кодировка ENC1 заменена на кодировку ENC2 для...)
В определении символьного шрифта использованы различные кодировки. Результатом может оказаться различие математических символов, которые печатает одна и та же команда (определённая при помощи `\DeclareMathSymbol`) в разных математических версиях (например, полужирной и нормальной насыщенности).

E `Encoding scheme` ENC `undefined.`
(Схема кодирования с именем ENC не известна.)
Объявите кодировку, указав ENC в необязательном аргументе `\usepackage` при загрузке пакете fontenc. Загрузка пакета fontenc должна предшествовать загрузке шрифтового пакета, вызвавшего ошибку. Если в исходном тексте документа используются шрифтовые команды низкого уровня типа `\fontencoding`, проверьте правильность написания ENC.

E `Environment` *env* `undefined.`
(Процедура *env* не определена.)
В аргументе команды `\begin` указано имя несуществующей процедуры или же имя процедуры пропущено.

i `External font` *font* `loaded for size` *size*`.`
(Внешний шрифт *font* загружен для размера *size*.)
Вместо запрошенного шрифта с размером *size* загружен шрифт *font* фиксированного размера согласно правилу подстановки, заданному в таблицах NFSS (в файлах определения шрифтов с расширением `fd`).

E `Extra alignment tab has been changed to \cr.`
(Лишний табулятор заменён на символ начала строки.)
В теле процедуры `array` или `tabular` число колонок в строке превысило число колонок, зарезервированных в аргументе процедур. Другими словами, слишком много символов `&` до конца строки. Возможно, пропущена команда `\\` в конце предшествующей строки.

E `Extra }, or forgotten $.`
(Лишняя `}` или забыт `$`.)
Нарушено соответствие открывающих и закрывающих фигурных скобок или команд перехода в математическую моду или из неё. Возможно, пропущено где-то `{`, `\[`, `\(` или `$`.

E `File` *file* `not found.`
(Файл *file* не найден.)
LaTeX пытается прочитать файл, который не существует. Если файл имеет расширение `tex`, тогда его пытаются прочитать команды `\input` или `\include`; если он имеет расширение `cls`, тогда указан несуществующий класс печатного документа в декларации `\documentclass`; если он имеет расширение `sty`, тогда указан несуществующий пакет в декларации `\usepackage`. LaTeX приостанавливает работу и ожидает ввода с клавиатуры правильного имени файла; при нажатии клавиши ⟨Enter⟩ работа будет продолжена без повторения попытки найти файл.

i `Float too large for page by` *size*`.`
(Плавающий объект больше страницы на *size*.)
Таблица или рисунок выходит за нижнюю границу страницы на размер, указанный в единицах pt.

i `FontDef file:` *file*`...`
(Файл определения шрифтов: *file*...)
Загружен файл с именем *file*, где задано соответствие команд переключения шрифтов конкретным внешним шрифтам.

E `Font family` ENC+*family* `unknown.`
(Семейство шрифтов ENC+*family* не известно.)
Обнаружена ошибка в шрифтовом пакете. Обратитесь к автору пакета.

E `Font` *name* `not found.`
(Шрифт *name* не найден.)
Ошибка в файлах определения шрифтов. Обратитесь к разработчику.

i `Font shape` *shape* `in size` *size* `not available.`
(Шрифт начертания *shape* размером *size* недоступен.)
Указанный шрифт отсутствует. Следующая строка сообщения покажет, какой шрифт будет использован для замещения отсутствующего. Если это сообщение вызвано декларацией `\boldmath`, отсутствующий шрифт, возможно, реально не используется в документе, так как `\boldmath` пытается одновременно загрузить

математические шрифты для основного размера, для индексов и индексов в индексах.

E `Font shape` *shape* `not found.`
(Шрифт с начертанием *name* не найден.)
Ошибка в файлах определения шрифтов. Обратитесь к разработчику.

i `Font shape` *shape* `will be scaled to size` *size*`.`
(Шрифт начертания *shape* будет отмасштабирован до размера *size*.)
Шрифт указанного начертания и размера будет получен из имеющегося путём увеличения или уменьшения до нужного размера. Современные программы печати документов умеют масштабировать любые используемые шрифты «на лету», поэтому данное предупреждение можно игнорировать.

i `Font shape` *shape1* `undefined. Using` *shape2* `instead.`
(Шрифт начертания *shape1* не определён. Взамен используется *shape2*.)
Система загрузки шрифтов приняла решение о подмене отсутствующего начертания другим. Правила подбора замещающих шрифтов определены в файлах определения шрифтов (с расширением `fd`).

E `I can't find file` *file*`.`
(Не найден файл с именем *file*.)
Возможно, выполнена попытка откомпилировать несуществующий входной файл. Эта ошибка может быть зафиксирована также, если пропущены фигурные скобки вокруг аргумента команды `\input`.

E `Illegal character in array arg.`
(Неправильный символ в аргументе процедуры `array`.)
Найден недопустимый символ в аргументе процедуры `array` или `tabular` или во втором аргументе команды `\multicolumn`.

E `Illegal parameter number in definition of` *command*`.`
(Неправильный номер параметра в определении *command*.)
Возможно, ошибка вызвана неправильным употреблением символа `#` в декларациях `\newcommand`, `\renewcommand`, `\newenvironment` или `\renewenvironment`. Символ `#` может быть использован только для определения параметров аргумента во вновь определяемой команде или процедуре или в имени команды `\#`. Например, `#2` означает аргумент номер 2. Эта ошибка также вызывается наличием параметра типа `#2` в последнем аргументе деклараций `\newenvironment` или `\renewenvironment`.

E `Illegal unit of measure (pt inserted).`
(Неправильная единица измерения (заменена на `pt`).)
Если непосредственно перед этой ошибкой получено сообщение `Missing number, treated as zero`, то это часть той же проблемы (см. ниже). Так или иначе LaTeX ожидал получить в качестве аргумента какой-то команды параметр длины, а вместо этого обнаружил число. Наиболее частой причиной ошибки является напи-

сание `0` вместо, например, `0in` при задании нулевой длины. Кроме того, ошибка может быть вызвана пропуском аргумента команды.

`Illegal use of \verb command.`
(Команда `\verb` использована в аргументе другой команды.)
Команду `\verb` нельзя использовать в аргументах других команд.

`Label` *key* `multiply defined.`
(Метка *key* определена повторно.)
Две команды `\label` или `\bibitem` имеют один и тот же аргумент.

`Label(s) may have changed. Rerun to get cross-references right.`
(Метки, возможно, были изменены. Повторите компиляцию, чтобы получить правильные перекрёстные ссылки.)
Перекрёстные ссылки, напечатанные командами `\ref`, `\pageref` или `\cite`, могут быть неправильными, так как они были изменены после предыдущей компиляции входного файла. Данное предупреждение печатается при завершении компиляции.

`LaTeX2e command` *command* `in LaTeX 2.09 document.`
(Команда *command* версии LaTeX 2_{ε} использована в документе LaTeX 2.09.)
Необходимо заменить её командой, существовавшей в версии LaTeX 2.09, либо выполнить компиляцию документа в режиме LaTeX 2_{ε}, как описано в приложении А.

`Lonely \item--perhaps a missing list environment.`
(Уединённая команда `\item` — возможно, пропущена процедура составления списка.)
Команда `\item` находится вне процедуры составления списка.

`Marginpar on page` *page* `moved.`
(Заметка на полях на странице *page* сдвинута.)
Заметка сдвинута вниз от строки, где появилась команда `\marginpar`, чтобы не печатать её поверх уже имеющейся заметки на полях.

`Math alphabet identifier` *alphabet* `is undefined in math version` *ver*`.`
(Идентификатор математического алфавита *alphabet* не определён в версии *ver*.)
Ошибка в шрифтовом пакете. Обратитесь к разработчику.

`Math version` *version* `is not defined.`
(Математическая версия *version* не определена.)
Ошибка в шрифтовом пакете. Обратитесь к разработчику.

`Misplaced alignment tab character &.`
(Лишний символ табуляции `&`.)
Символ `&` использован в обычном тексте. Он может использоваться в процедурах `array` и `tabular`. Возможно, следовало напечатать `\&`.

`Missing \begin{document}.`
(Пропущена команда `\begin{document}`.)

Текст или команда, печатающая текст, предшествуют команде `\begin{document}`. Возможно, она пропущена или содержится ошибка в преамбуле, например в одной из деклараций пропущены фигурные скобки вокруг аргумента или пропущен обратный слеш `\` в имени декларации. Возможно, вместо корневого входного файла предпринята попытка выполнить компиляцию файла, подключаемого к корневому файлу при помощи команды `\input` или `\include`.

E `Missing p-arg in array arg.`
(Пропущен `p`-аргумент в аргументе матрицы.)
В аргументе процедуры `array` или `tabular` или во втором аргументе команды `multicolumn` имеется `p` (признак форматирования в парбоксе), за которым пропущено выражение в фигурных скобках.

E `Missing @-exp in array arg.`
(Пропущено `@`-выражение в аргументе матрицы.)
В аргументе процедуры `array` или `tabular` или во втором аргументе команды `\multicolumn` имеется символ `@`, за которым не следует `@`-выражение.

i `Missing character: There is no` *glyph* `in font` *font*`!`
(Пропущенный символ: нет символа *glyph* в шрифте *font*!)
В указанном шрифте отсутствует указанный символ.

E `Missing control sequence inserted.`
(Вставлена пропущенная команда.)
Возможно, первый аргумент `\newcommand`, `\renewcommand` или `\newlength` не является именем команды, например, в имени команды пропущен обратный слеш.

E `Missing { inserted.`
(Пропущенная фигурная скобка `{` вставлена.)
`Missing } inserted.`
(Пропущенная фигурная скобка `}` вставлена.)
TeX не понимает, что следует делать. Возможно, неверный ввод во входном файле находится ранее того места, где была зафиксирована данная ошибка.

E `Missing $ inserted.`
(Пропущенный символ `$` вставлен.)
TeX, возможно, нашёл команду, которая может быть использована только в математической моде, а стоит она не в этой моде. Следует помнить, что TeX действует не в математической моде, когда он обрабатывает аргумент команды, формирующей бокс, даже когда эта команда стоит внутри математической процедуры. Эта ошибка также возникает, когда TeX встречает пустую строку, находясь в математической моде.

E `Missing number, treated as zero.`
(Пропущено число, заменено на ноль.)
TeX, возможно, исполнял команду LaTeX'а, ожидавшую в качестве аргумента параметр длины и не получившую его. Возможно, пропущен аргумент или квадратная скобка в начале или конце необязательного аргумента. Эту ошибку также

вызывает вставка команды `\protect` либо перед командной длиной, либо перед командой `\value`, которая производит число.

`NFSS release 1 command` *name* `found.`
(Обнаружена команда *name* первого выпуска NFSS.)
Это сообщение может быть как предупреждением, так и информацией об ошибке. Команда с именем *name* использовалась в первом выпуске новой схемы выбора шрифтов (New Font Selection Scheme), но позднее была заменена другой. В большинстве случаев компилятор способен подставить вместо устаревшей команды её более новый аналог, однако в любом случае необходимо внести исправления в исходный текст документа или получить более новую версию пакета, содержащего команду *name*.

`No \author given.`
(Не задана `\author`.)
Команде `\maketitle` не предшествует `\author`.

`No counter` *counter* `defined.`
(Счётчик *counter* не определён.)
Использовано несуществующее имя счётчика в обязательном аргументе команды `\setcounter` или `\addtocounter` или в необязательном аргументе деклараций `\newcounter` или `\newtheorem`. Однако, если эта ошибка обнаружена во время чтения `aux`-файла, тогда, возможно, она вызвана использованием декларации `\newcounter` в файле, который включен в документ посредством команды `\include`.

`No declaration for shape` *shape*.
(Нет декларации для начертания *shape*.)
Выполнена подстановка шрифта, для которого не определено начертание *shape*.

`No \title given.`
(Не задан `\title`.)
Команде `\maketitle` не предшествует `\title`.

`Not a letter.`
(Не буква.)
В аргументе команды `\hyphenation` имеется что-то, чего там быть не должно.

`Not in outer par mode.`
(Не во внешней текстовой моде.)
Использована процедура `figure` или `table` или команда `\marginpar` в математической моде или в аргументе команды `\parbox`. Плавающий объект может быть создан только во внешней текстовой моде.

`Option clash for package` *package*.
(Конфликт опций для пакета *package*.)
Указанный пакет загружен дважды с разными опциями. Нажав ⟨h⟩⟨Enter⟩, можно получить список опций при каждой загрузке. Выход состоит в том, чтобы

загрузить пакет сразу со всеми необходимыми опциями. Следует иметь в виду, что пакеты могут загружать другие пакеты, поэтому указанную ошибку можно получить без явного вызова дважды одного и того же пакета.

`Optional argument of \twocolumn too tall on page` *page*.
(Необязательный аргумент команды `\twocolumn` на стр. *page* слишком большой.)
Значение необязательного аргумента команды `\twocolumn` превышает высоту бокса, который мог бы уместиться на странице.

`Oval too small.`
(Слишком маленький овал.)
Команда `\oval` определяет овал настолько малым, что LaTeX не может вписать четверть окружности в соответствующий угол.

`Overfull \hbox ...`
(Переполнен горизонтальный бокс.)
Этот случай описан в разделе 4.4.

`Overfull \vbox ...`
(Переполнен вертикальный бокс.)
Не было найдено подходящего места для прерывания страницы. TeX сформировал страницу большей длины, чем определено классом печатного документа. Как формировать страницу, рассказывает раздел 4.7.

`Overwriting` *font* `in version` *version*.
(Переопределён *font* в версии *version*.)
Изменено начертание символьного шрифта или математического алфавита *font* в версии *version*.

`Paragraph ended before` *command* `was complete.`
(Абзац закончился прежде, чем завершено выполнение команды *command*.)
Обнаружена пустая строка в аргументе команды *command*, где не должно быть пустых строк. Возможно, пропущена правая фигурная скобка в конце аргумента указанной команды.

`Redeclaring math alphabet` *alphabet*.
(Переопределён математический алфавит *alphabet*.)
Изменён шрифт, используемый математическим алфавитом *alphabet*. Возможно, изменение произведено загруженным пакетом.

`Redeclaring math symbol` *symbol*.
(Переопределён математический символ *symbol*.)
Изменена команда *symbol*, печатающая один из математических символов. Возможно, изменение произведено загруженным пакетом.

`Redeclaring math version` *version*.
(Переопределена математическая версия *version*.)
Изменена версия *version* математических шрифтов. Возможно, изменение произведено загруженным пакетом.

`Redeclaring symbol font` *name*`.`
(Переопределён символьный шрифт *name*.)
Изменён символьный шрифт с именем *name* во всех математических версиях. Возможно, изменение произведено загруженным пакетом.

`Reference` *key* `on page` *page* `undefined.`
(Ссылка *key* на странице *page* не определена.)
Аргумент команды `\ref` или `\pageref` не определён командой `\label`.

`Size substitution with differences to` *size* `have occurred.`
(Произведена подстановка шрифтов с несоответствием размеров до *size*.)
Обнаружена хотя бы одна подмена шрифтов с существенной разницей в размерах. Величина *size* обозначает максимальную разницу для всего множества произведённых подмен.

`Some font shapes were not available, defaults substituted.`
(Шрифты некоторых начертаний недоступны, заменены используемыми по умолчанию.)
Данное сообщение выводится в конце компиляции документа, если производилась автоматическая подстановка отсутствующих шрифтов.

`Something wrong--perhaps a missing \item.`
(Что-то не в порядке — возможно, пропущена команда `\item`.)
Наиболее вероятная причина этой ошибки — пропуск команды `\item` в теле процедуры составления списков или пропуск аргумента такой процедуры, как `thebibliography`.

`Symbol font` *name* `is not defined.`
(Символьный шрифт *name* не определён.)
Ошибка в шрифтовом пакете. Обратитесь к автору.

`Tab overflow.`
(Переполнение табуляторов.)
Число команд `\=` превосходит максимальное число допустимых в LaTeX'е табуляторов.

`TeX capacity exceeded, sorry [...]`
(Возможности TeX'а превышены, извините `[...]`)
TeX вышел за отведённую ему область памяти и прекратил работу. Прежде чем паниковать, следует подумать, что могло вызвать эту ошибку. Ниже объясняется, как определить, действительно ли выделенный объём памяти превышен, и что делать, если памяти действительно не хватает. При отсутствии ошибок во входном файле компилятор редко выходит за отведённые пределы памяти, поэтому разбиение документа на короткие части часто помогает локализовать ошибку, послужившую причиной столь печального сообщения. Если ошибки в исходном тексте не найдены, следует проверить, что операционная система способна предоставить компилятору достаточный объём виртуальной памяти. Например,

в операционной системе Windows при небольшом объёме памяти оперативного запоминающего устройства (ОЗУ) необходимо убедиться, что жёсткий диск компьютера имеет достаточно свободного места, чтобы Windows могла увеличить размер файла подкачки.

Конец строки индикации ошибок, который выше обозначен как `[...]`, указывает, памяти какого типа не хватило TEX'у. Наиболее общие варианты перечислены ниже.

`buffer memory size` (размер буфера памяти).
: Ошибка может быть вызвана слишком длинным фрагментом текста в аргументе команд секционирования, команды `\caption`, `\addcontentsline` или `\addtocontents`. Эта ошибка обнаруживается при обработке команды `\end {document}`, но также может быть зафиксирована при исполнении команд `\tableofcontents`, `\listoffigures` или `\listoftables`. Чтобы избежать этой ошибки, следует записывать в необязательный аргумент команд секционирования и команды `\caption` соответственно сокращённый вариант заголовков разделов и подписей к плавающим объектам (рисункам и таблицам). Длинные названия в оглавлении или списках рисунков и таблиц вряд ли целесообразны.

`exception dictionary` (словарь исключений).
: Следует уменьшить объём информации о переносах слов, вводимой с помощью деклараций `\hyphenation`, за счёт редко используемых слов. Правило переноса этих слов можно указать с помощью команды `\-` непосредственно там, где эти слова встречаются во входном файле.

`hash size` (хэш-размер).
: Во входном файле введено слишком много новых команд и/или используется слишком много меток для перекрёстных ссылок.

`input stack size` (размер стека).
: Причиной, возможно, является ошибка при определении новой команды. Например, следующая команда ошибочна, так как определяет команду `\false` циклически:

```
\newcommand{\false}{$\false$}
```

Когда TEX встретит команду `\false` во входном файле, он будет пытаться раскрыть её определение до уровня базисных команд. Очевидно, что в примере с командой `\false` этого никогда не случится, а в результате вся доступная память будет исчерпана.

`main memory size` (размер оперативной памяти).
: Возможны три случая, когда TEX'у не хватает оперативной памяти:

(1) определено много сложных команд и/или процедур;

(2) при составлении алфавитного указателя слишком много команд `\index` или `\glossary` попадают на одну страницу;

(3) сформатирована настолько сложная страница, что TEX не способен удержать её содержание в своей памяти.

Решение двух первых проблем очевидно — нужно определить меньше команд или использовать меньше команд `\index` и `\glossary`. Что касается третьей причины, то, возможно, вывод на одну страницу производится слишком большими процедурами `tabbing`, `tabular`, `array` и `picture`. Память может быть также переполнена рисунками и таблицами, ждущими их размещения. Чтобы определить, действительно ли переполнена отведённая TEX'у область памяти, можно вставить команду `\clearpage` во входной файл непосредственно перед тем местом, где TEX выходит за отведённую ему область, и обработать входной файл снова. Если теперь ошибки нет, то действительно причиной ошибки было переполнение указанного вида памяти. Если же переполнение продолжается, то, вероятно, имеется ошибка во входном файле.

Что же делать, если TEX действительно выходит за рамки отведённого объёма памяти? TEX всегда производит обработку полного абзаца перед тем, как решить, где начать новую страницу. Вставляя команду `\newpage`, можно заставить TEX сформировать страницу до завершения обработки текущего абзаца, часть которого попадает на другую страницу. Если ошибка возникает из-за накопления фигур и таблиц, можно попытаться воспрепятствовать этому, рассредоточив их дальше по тексту (глава 11).

`pool size` (размер пула).

Возможно, используется слишком много меток для перекрёстных ссылок и (или) определено слишком много новых команд. Если говорить более точно, метки и имена команд содержат слишком много букв; следует попробовать использовать более короткие имена. Однако чаще всего такая ошибка вызвана пропуском правой скобки аргумента таких команд, как `\setcounter`, `\newtheorem` или `\newenvironment`.

`save size` (размер хранения).

Эта ошибка случается, когда команды, процедуры или декларации вложены слишком глубоко друг в друга: например, аргумент команды `\multiput` содержит процедуру `picture`, которая, в свою очередь, содержит декларацию `\footnotesize`, в области действия которой вновь имеется команда `\multiput`, содержащая ещё что-нибудь, и т. д.

E `Text line contains an invalid character.`
(Строка текста содержит неправильный символ.)
Входной файл содержит символы с кодами, отсутствующими в указанной кодировке исходного текста. Возможно, указана неверная кодировка, например `cp866` (MS DOS) вместо `cp1251` (MS Windows). Загрузите пакет `inputenc` с опцией, соответствующей кодировке исходного текста документа.

E `Text for \verb command ended by end of line.`
(Команда `\verb` начата, но не завершена на текущей строке.)

Следует проверить, не забыт ли второй символ, метящий границы аргумента команды `\verb`.

`There were multiply-defined labels.`
(Были многократно определённые метки.)
Данное предупреждение печатается при завершении компиляции, если две команды `\label` использовали одну и ту же метку.

`There were undefined references or citations.`
(Были неопределённые метки или ссылки на литературу.)
Данное предупреждение печатается при завершении компиляции, если были найдены команды `\ref` или `\cite`, которые не имеют соответствующих `\label` или `\bibitem`.

`There's no line here to end.`
(Далее нет строки.)
Команда `\newline` или `\\` находится между абзацами, где она бессмысленна. Следует использовать команду `\vspace`.

`Try loading font information for` ENC+*family*`.`
(Поиск информации для загрузки шрифтов семейства *family* в кодировке ENC.)
Данное предупреждение записывается в файл протокола во всех случаях, когда компилятор пытается загрузить файл определения шрифтов для заданной комбинации. Файл имеет расширение `fd`, а его имя получается склеиванием слов ENC и *family*.

`This may be a LaTeX bug.`
(Неидентифицируемая ошибка.)
Не исключено, что она вызвана предыдущими ошибками. Если сообщению об этой ошибке не предшествуют другие ошибки, возможно, что обнаружена ошибка в самом LaTeX'е; тогда следует обратиться к эксперту.

`This NFSS system isn't set up properly.`
(Система NFSS настроена неправильно.)
В системе загрузки шрифтов обнаружена фатальная ошибка. Ошибка возникает, если в конце логической цепочки подстановки отсутствующих шрифтов оказывается также отсутствующий шрифт. Переустановите LaTeX или обратитесь к разработчику шрифтового пакета, вызвавшего ошибку.

`Too deeply nested.`
(Слишком глубокое вложение.)
Слишком много списков вложено друг в друга. Предельный уровень вложенности зависит от выбора класса печатного документа, однако, по крайней мере, четыре уровня гарантированы; обычно этого вполне достаточно.

`Too many columns in eqnarray environment.`
(Слишком много колонок в процедуре `eqnarray`.)

Процедура `eqnarray` содержит три разделителя колонок `&` без промежуточной команды `\\` перехода на новую строку.

E `Too many unprocessed floats.`
(Слишком много необработанных плавающих объектов.)
Превышен объём памяти, зарезервированной для накопления плавающих объектов. Возможно, использовано слишком большое количество процедур `table`, `figure` и команд `\marginpar`, ждущих очереди для размещения на момент завершения форматирования очередной страницы. Возможно, имеется плавающий объект, который, в соответствии с его параметрами, может быть размещён только в конце печатного документа, и за ним скопилась очередь из других плавающих объектов. Возможно, следует добавить опцию `p` в необязательный аргумент соответствующей процедуры `table` или `figure` или перераспределить расположение объектов во входном файле. Если внесение изменений в исходный текст нежелательно, попробуйте увеличить размер виртуальной памяти, выделенной компилятору.

E `Undefined size function` *name*.
(Шрифтовая размерная функция *name* не определена.)
Ошибка в шрифтовом пакете. Обратитесь к автору пакета.

E `Undefined control sequence.`
(Неопределённая команда.)
TeX не может опознать команду. Возможно, допущена опечатка в её имени. Если TeX зафиксировал данную ошибку при исполнении команды LaTeX'а, причиной ошибки может быть использование команды в неправильном контексте; например, команда `\item` находится вне тела процедуры, форматирующей список. Другой причиной может быть пропуск деклараций `\documentclass` или `\usepackage`.

E `Undefined tab position.`
(Неопределённая позиция табулятора.)
В теле процедуры `tabbing` использована команда `\>`, `\+`, `\-` или `\<` для перехода к несуществующей позиции табуляции, которая не была определена ранее соответствующей командой `\=`.

i `Underfull \hbox ...`
(Не заполнен горизонтальный бокс.)
Проверьте печатный документ на наличие больших вертикальных пробелов. Возможно, они связаны с командами `\\` или `\newline`, например двумя подряд командами `\\`. Это предупреждение может быть вызвано также использованием процедуры `sloppypar`, декларации `\sloppy` или команды `\linebreak`.

i `Underfull \vbox ...`
(Не заполнен вертикальный бокс.)
TeX не нашёл подходящего места для прерывания страницы, и на странице осталось незаполненное пространство (раздел 4.7).

E `Use of` *command* `doesn't match its definition.`
(Использование *command* не соответствует её определению.)
Если в *command* указана одна из команд рисования, следует проверить синтаксис её аргументов (он отличается от синтаксиса других команд). Если в *command* указана команда `\@array`, то ошибка находится в `@`-выражении в аргументе процедуры `array` или `tabular`; в частности, хрупкие команды в `@`-выражении должны быть защищены командой `\protect`. Данная ошибка может быть также вызвана незащищённой (при помощи `\protect`) командой с необязательным аргументом в подвижном аргументе другой команды (причём в *command* может быть указано имя совсем другой команды).

E `Unknown option` *option* `for package` *package*`.`
(Неизвестная опция *option* для пакета *package*.)
Проверьте необязательный аргумент декларации `\usepackage`.

E `You can't use macro parameter character # in` *mode* `mode.`
(Вы не можете использовать символ макропараметра `#` в моде *mode*.)
Спецсимвол `#` обнаружен в обычном тексте. Возможно, следовало напечатать `\#`.

i `You have requested release` *date1* `of` *package*`, but only release` *date2* `is available.`
(Затребована версия *package*, датированная *date1*, но доступна только версия от *date2*.)
Получите более новую версию класса или пакета *package*, вызвавшего это сообщение.

B.4. Диагностические сообщения *MakeIndex*

MakeIndex выводит на экран число прочитанных и записанных строк и количество обнаруженных ошибок. Сообщения об ошибках записываются в файл с расширением `ilg`. *MakeIndex* может обнаружить ошибки на стадии чтения, когда он считывает файл с расширением `idx`, и на стадии записи, когда он записывает файл с расширением `ind`. Каждое сообщение указывает признак ошибки и номер строки в файле, где она найдена. На стадии чтения номер строки относится к `idx`-файлу; на стадии записи — к `ind`-файлу.

B.4.1. Ошибки на стадии чтения

`Extra ! at position...`
(Лишний `!` в позиции...)
Аргумент команды `\index` содержит три или более символа `!`, которым не предшествуют двойные кавычки `"`. Аналогичная ошибка регистрируется при наличии двух или более символов `@` и `|` без кавычек. Возможно, некоторые из них нужно процитировать при помощи `"`.

`Extra @ at position...`
См. выше.

`Extra | at position...`
См. выше.

`Illegal null field.`
(Неверное пустое поле.)
Аргумент команды `\index` не имеет смысла. Например, `\index{!big}` приводит к этой ошибке, поскольку указывает вход второго уровня «big» для пустого входа первого уровня. Аналогично команда `\index{@big}` ошибочна, потому что указывает пустую строку для расстановки по алфавиту.

`No \indexentry.`
(Нет входа.)
Каждая строка `idx`-файла, записанная LaTeX'ом, должна содержать команду `\indexentry`. *MakeIndex* обнаружил строку в `idx`-файле, которая не содержит такой команды. Возможно, файл испорчен.

`Missing {.`
(Пропущена фигурная скобка `{`.)
MakeIndex считает, что в аргументе команды `\index` имеются непарные фигурные скобки. Возможно, `idx`-файл испорчен.

`Missing }.`
См. выше.

`Argument ... too long.`
(Аргумент `...` слишком длинный.)
Обнаружена команда `\index` с очень длинным аргументом. Возможно, пропущена правая фигурная скобка, которая ограничивает аргумент.

B.4.2. Ошибки на стадии записи

`Unmatched range opening operator.`
(Непарный оператор начала диапазона.)
За командой `\index{...|(}` не найдена парная ей команда конца диапазона `\index{...|)}`. Часть аргумента, обозначенная многоточием, должна быть полностью идентична в обеих командах.

`Unmatched range closing operator.`
(Непарный оператор конца диапазона.)
Перед командой `\index{...|)}` не найдена парная ей команда начала диапазона `\index{...|(}`. Часть аргумента, обозначенная многоточием, должна быть полностью идентична в обеих командах.

`Extra range opening operator.`
(Лишний оператор начала диапазона.)

Обнаружены подряд две команды `\index{...|(}` с идентичным аргументом без необходимой в этом случае промежуточной команды `\index{...|)}`.

`Inconsistent page encapsulator ... within range.`
(Несогласованное описание страницы ... внутри диапазона.)
MakeIndex получил инструкцию включить в указатель диапазон страниц и отдельную страницу внутри этого диапазона, сформатировав её номер иным способом. Например, команда `\index{гну|textit}` расположена между командами `\index{гну|(}` и `\index{гну|)}`.

`Conflicting entries.`
(Конфликтующие входы.)
MakeIndex считает, что получил инструкцию напечатать номер одной и той же страницы для одного и того же входа дважды двумя разными способами. Например, команды `\index{гну}` и `\index{гну|see{...}}` попали на одну страницу.

В.4.3. Другие ошибки

MakeIndex может регистрировать ряд других ошибок, указывающих на нечто существенно неверное в `idx`-файле. Если такая ошибка зарегистрирована, возможно, что файл испорчен. Однако, если LaTeX не зафиксировал какой-либо ошибки при его записи, тогда почти наверняка он создал нормальный `idx`-файл. В таком случае его необходимо исследовать, чтобы выяснить, в чём дело.

«Шифровка!» — подумал Мюллер.
«UUENCODE» — подумал Штирлиц.
http://lib.ru. Анекдоты про Штирлица

Приложение C

Таблицы кодировок

Таблица C.1

Кодировка `OT1` (все шрифты, кроме машинописных)

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
"0x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	Φ	Ψ	Ω	ff	fi	fl	ffi	ffl
"1x	ı	ȷ	`	´	ˇ	˘	¯	˚	¸	ß	æ	œ	ø	Æ	Œ	Ø
"2x	˗	!	”	#	$	%	&	’	(	)	*	+	,	-	.	/
"3x	0	1	2	3	4	5	6	7	8	9	:	;	¡	=	¿	?
"4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
"5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	“	]	ˆ	˙
"6x	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
"7x	p	q	r	s	t	u	v	w	x	y	z	–	—	˝	˜	¨

Таблица C.2

Кодировка `OT1` (машинописные шрифты)

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
"0x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	Φ	Ψ	Ω	↑	↓	'	¡	¿
"1x	ı	ȷ	`	´	ˇ	˘	¯	˚	¸	ß	æ	œ	ø	Æ	Œ	Ø
"2x	␣	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
"3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
"4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
"5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
"6x	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
"7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	¨

Таблица C.3

Кодировка `T1`

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
"0x	`	´	ˆ	˜	¨	˝	˚	ˇ	˘	¯	˙	¸	˛	‚	‹	›
"1x	“	”	„	«	»	–	—		₀	ı	ȷ	ff	fi	fl	ffi	ffl
"2x	␣	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
"3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
"4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
"5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
"6x	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
"7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	-
"8x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
"9x	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§
"Ax	ă	ą	ć	č	ď	ě	ę	ğ	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
"Bx	ř	ś	š	ş	ť	ţ	ű	ů	ÿ	ź	ž	ż	ĳ	¡	¿	£
"Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
"Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS
"Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
"Fx	ð	ñ	ò	ó	ô	õ	ö	œ	ø	ù	ú	û	ü	ý	þ	ß

Таблица C.4

Кодировка `TS1`

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
"0x	`	´	ˆ	˜	¨	˝	˚	ˇ	˘	¯	˙	¸	˛	‚		
"1x			"			—	—		←	→	⁀	⁀	⁀	⁀		
"2x	␢				$			'			*		,	=	.	/
"3x	0	1	2	3	4	5	6	7	8	9			〈	—	〉	
"4x														℧		◯
"5x								Ω				⟦		⟧	↑	↓
"6x	`		★	⚮	†								🙘	⚭	♪	
"7x															~	=
"8x	˘	ˇ	˝	‶	†	‡	‖	‰	•	℃	$	¢	ƒ	₡	₩	₦
"9x	₲	₱	₤	℞	‽	⸘	₫	™	‱	¶	฿	№	⁒	℮	º	℠
"Ax	{	}	¢	£	¤	¥	¦	§	¨	©	ª	🄯	¬	℗	®	¯
"Bx	°	±	²	³	´	µ	¶	·	※	¹	º	√	¼	½	¾	€
"Cx																
"Dx							×									
"Ex																
"Fx							÷									

Таблица C.5

Кодировка `T2A`

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
`"0`x	`	´	ˆ	˜	¨	˝	˚	ˇ	˘	¯	˙	¸	˛	I	⟨	⟩
`"1`x	“	”	⌢	‟	̑	–	—		₀	ı	ȷ	ff	fi	fl	ffi	ffl
`"2`x	␣	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
`"3`x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
`"4`x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
`"5`x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	ˆ	_
`"6`x	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
`"7`x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	˜	-
`"8`x	Ґ	Ғ	Ђ	Ћ	Һ	Җ	Ҙ	Љ	Ї	Қ	Ҡ	Ҝ	Ӕ	Ң	Ҥ	Ѕ
`"9`x	Ө	Ҫ	Ў	Ү	Ұ	Ҳ	Џ	Ҹ	Ҷ	Є	Ә	Њ	Ё	№	¤	§
`"A`x	ґ	ғ	ђ	ћ	һ	җ	ҙ	љ	ї	қ	ҡ	ҝ	ӕ	ң	ҥ	ѕ
`"B`x	ө	ҫ	ў	ү	ұ	ҳ	џ	ҹ	ҷ	є	ә	њ	ё	„	«	»
`"C`x	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
`"D`x	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
`"E`x	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
`"F`x	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

Таблица C.6

Кодировка `OML`

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
`"0`x	$\mathit{\Gamma}$	$\mathit{\Delta}$	$\mathit{\Theta}$	$\mathit{\Lambda}$	$\mathit{\Xi}$	$\mathit{\Pi}$	$\mathit{\Sigma}$	$\mathit{\Upsilon}$	$\mathit{\Phi}$	$\mathit{\Psi}$	$\mathit{\Omega}$	α	β	γ	δ	ϵ
`"1`x	ζ	η	θ	ι	κ	λ	μ	ν	ξ	π	ρ	σ	τ	υ	ϕ	χ
`"2`x	ψ	ω	ε	ϑ	ϖ	ϱ	ς	φ	$\leftharpoonup$	$\leftharpoondown$	$\rightharpoonup$	$\rightharpoondown$	$\lhook$	$\rhook$	$\triangleright$	$\triangleleft$
`"3`x	0	1	2	3	4	5	6	7	8	9	.	,	$<$	$/$	$>$	$\star$
`"4`x	∂	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
`"5`x	P	Q	R	S	T	U	V	W	X	Y	Z	$\flat$	$\natural$	$\sharp$	$\smile$	$\frown$
`"6`x	ℓ	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
`"7`x	p	q	r	s	t	u	v	w	x	y	z	$\imath$	$\jmath$	$\wp$	$\vec{}$	⁀

Таблица C.7

Кодировка `OMS`

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
"0x	$-$	$\cdot$	$\times$	$\ast$	$\div$	$\diamond$	$\pm$	$\mp$	$\oplus$	$\ominus$	$\otimes$	$\oslash$	$\odot$	$\bigcirc$	$\circ$	$\bullet$
"1x	$\asymp$	$\equiv$	$\subseteq$	$\supseteq$	$\leq$	$\geq$	$\preceq$	$\succeq$	$\sim$	$\approx$	$\subset$	$\supset$	$\ll$	$\gg$	$\prec$	$\succ$
"2x	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$	$\leftrightarrow$	$\nearrow$	$\searrow$	$\simeq$	$\Leftarrow$	$\Rightarrow$	$\Uparrow$	$\Downarrow$	$\Leftrightarrow$	$\nwarrow$	$\swarrow$	$\propto$
"3x	$\prime$	∞	$\in$	$\ni$	$\bigtriangleup$	$\bigtriangledown$	$/$	$\mapstochar$	$\forall$	$\exists$	$\neg$	$\emptyset$	$\Re$	$\Im$	$\top$	$\bot$
"4x	$\aleph$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$	$\mathcal{H}$	$\mathcal{I}$	$\mathcal{J}$	$\mathcal{K}$	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{N}$	$\mathcal{O}$
"5x	$\mathcal{P}$	$\mathcal{Q}$	$\mathcal{R}$	$\mathcal{S}$	$\mathcal{T}$	$\mathcal{U}$	$\mathcal{V}$	$\mathcal{W}$	$\mathcal{X}$	$\mathcal{Y}$	$\mathcal{Z}$	$\cup$	$\cap$	$\uplus$	$\wedge$	$\vee$
"6x	$\vdash$	$\dashv$	$\lfloor$	$\rfloor$	$\lceil$	$\rceil$	$\{$	$\}$	$\langle$	$\rangle$	$\vert$	$\Vert$	$\updownarrow$	$\Updownarrow$	$\backslash$	$\wr$
"7x	$\surd$	$\amalg$	∇	$\smallint$	$\sqcup$	$\sqcap$	$\sqsubseteq$	$\sqsupseteq$	$\S$	$\dagger$	$\ddagger$	$\P$	$\clubsuit$	$\diamondsuit$	$\heartsuit$	$\spadesuit$

Таблица C.8

Кодировка `OMX`

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
"0x	$($	$)$	$[$	$]$	$\lfloor$	$\rfloor$	$\lceil$	$\rceil$	$\{$	$\}$	$\langle$	$\rangle$	$\vert$	$\Vert$	$/$	$\backslash$
"1x	$\big($	$\big)$	$\Big($	$\Big)$	$\Big[$	$\Big]$	$\Big\lfloor$	$\Big\rfloor$	$\Big\lceil$	$\Big\rceil$	$\Big\{$	$\Big\}$	$\Big\langle$	$\Big\rangle$	$\Big/$	$\Big\backslash$
"2x	$\Bigg($	$\Bigg)$	$\Bigg[$	$\Bigg]$	$\Bigg\lfloor$	$\Bigg\rfloor$	$\Bigg\lceil$	$\Bigg\rceil$	$\Bigg\{$	$\Bigg\}$	$\Bigg\langle$	$\Bigg\rangle$	$\Bigg/$	$\Bigg\backslash$	$\bigg/$	$\bigg\backslash$
"3x	⎛	⎞	⎡	⎤	⎣	⎦	⎢	⎥	⎧	⎫	⎩	⎭	⎨	⎬	⎪	$\vert$
"4x	⎝	⎠	⎜	⎟	$\bigg\langle$	$\bigg\rangle$	$\bigsqcup$	$\displaystyle\bigsqcup$	$\oint$	$\displaystyle\oint$	$\bigodot$	$\displaystyle\bigodot$	$\bigoplus$	$\displaystyle\bigoplus$	$\bigotimes$	$\displaystyle\bigotimes$
"5x	$\sum$	$\prod$	$\int$	$\bigcup$	$\bigcap$	$\biguplus$	$\bigwedge$	$\bigvee$	$\displaystyle\sum$	$\displaystyle\prod$	$\displaystyle\int$	$\displaystyle\bigcup$	$\displaystyle\bigcap$	$\displaystyle\biguplus$	$\displaystyle\bigwedge$	$\displaystyle\bigvee$
"6x	$\coprod$	$\displaystyle\coprod$	$\widehat{\ }$	$\widehat{\ \ }$	$\widehat{\ \ \ }$	$\widetilde{\ }$	$\widetilde{\ \ }$	$\widetilde{\ \ \ }$	$\big[$	$\big]$	$\big\lfloor$	$\big\rfloor$	$\big\lceil$	$\big\rceil$	$\big\{$	$\big\}$
"7x	$\surd$	$\big\surd$	$\Big\surd$	$\bigg\surd$	⎷	$\vert$	⌈	$\Vert$	$\uparrow$	$\downarrow$	⌜	⌝	⌞	⌟	$\Uparrow$	$\Downarrow$

Таблица С.9

Шрифт Symbol (кодировка U, гарнитура psy)

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
"0x																
"1x																
"2x		!	∀	#	∃	%	&	∋	(	)	∗	+	,	−	.	/
"3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
"4x	≅	Α	Β	Χ	Δ	Ε	Φ	Γ	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο
"5x	Π	Θ	Ρ	Σ	Τ	ϒ	ς	Ω	Ξ	Ψ	Ζ	[	∴	]	⊥	_
"6x	‾	α	β	χ	δ	ε	ϕ	γ	η	ι	φ	κ	λ	μ	ν	ο
"7x	π	θ	ρ	σ	τ	υ	ϖ	ω	ξ	ψ	ζ	{	|	}	~	
"8x																
"9x																
"Ax		ϒ	′	≤	⁄	∞	ƒ	♣	♦	♥	♠	↔	←	↑	→	↓
"Bx	°	±	″	≥	×	∝	∂	•	÷	≠	≡	≈	…	⏐	⎯	↵
"Cx	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
"Dx	∠	∇	®	©	™	∏	√	⋅	¬	∧	∨	⇔	⇐	⇑	⇒	⇓
"Ex	◊	〈	®	©	™	∑	⎛	⎜	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪
"Fx		〉	∫	⌠	⎮	⌡	⎞	⎟	⎠	⎤	⎥	⎦	⎫	⎬	⎭	

Таблица С.10

Шрифт ZapfDingbats (кодировка U, гарнитура pzd)

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
"0x																
"1x																
"2x		✁	✂	✃	✄	☎	✆	✇	✈	✉	☛	☞	✌	✍	✎	✏
"3x	✐	✑	✒	✓	✔	✕	✖	✗	✘	✙	✚	✛	✜	✝	✞	✟
"4x	✠	✡	✢	✣	✤	✥	✦	✧	★	✩	✪	✫	✬	✭	✮	✯
"5x	✰	✱	✲	✳	✴	✵	✶	✷	✸	✹	✺	✻	✼	✽	✾	✿
"6x	❀	❁	❂	❃	❄	❅	❆	❇	❈	❉	❊	❋	●	❍	■	❏
"7x	❐	❑	❒	▲	▼	◆	❖	◗	❘	❙	❚	❛	❜	❝	❞	
"8x																
"9x																
"Ax		❡	❢	❣	❤	❥	❦	❧	♣	♦	♥	♠	①	②	③	④
"Bx	⑤	⑥	⑦	⑧	⑨	⑩	❶	❷	❸	❹	❺	❻	❼	❽	❾	❿
"Cx	➀	➁	➂	➃	➄	➅	➆	➇	➈	➉	➊	➋	➌	➍	➎	➏
"Dx	➐	➑	➒	➓	➔	→	↔	↕	➘	➙	➚	➛	➜	➝	➞	➟
"Ex	➠	➡	➢	➣	➤	➥	➦	➧	➨	➩	➪	➫	➬	➭	➮	➯
"Fx		➱	➲	➳	➴	➵	➶	➷	➸	➹	➺	➻	➼	➽	➾	

Предметный указатель

Курсивом выделены номера страниц с определениями команд, процедур или терминов. В списке пакетов и классов подчеркнуты номера страниц с определениями, специфичными для соответствующего пакета или класса.

СИМВОЛЫ

C

D

I

J

K

L

M

Q

R

S

U

Л

Р

С

Т

У

Ф

Я

Именной указатель

Литература

[1] *Donald E. Knuth.* The Art of Programming. Reading, Massachusetts: Addison-Wesley Publishing Company, 1973.

Имеется перевод: *Д. Кнут.* Искусство программирования для ЭВМ. В 3 т. М.: Мир, 1976–1978.

[2] *Donald E. Knuth.* The TEXbook. Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

Имеется перевод: *Д. Е. Кнут.* Всё про TEX. Протвино: Изд-во АО RDTEX, 1993.

[3] *Donald E. Knuth.* The METAFONTbook. Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

[4] *Douglas Downing and Michael A. Covington.* Dictionary of Computer and Internet Terms. 8th ed. Barron's Educational Series. New York: Woodbury, 2003.

[5] *Leslie Lamport.* LATEX: A Document Preparation System. 2nd ed. Reading, Massachusetts: Addison-Wesley, 1994.

[6] *Michael Spivak.* Joy of Tex: A Gourmet Guide to Typesetting With the AMS-TEX Macro Package. Providence, RI: American Mathematical Society, 1990.

Имееется перевод: *М. Спивак.* Восхитительный TEX: Руководство по комфортному изготовлению научных публикаций в пакете AMS-TEX. М.: Мир, 1993.

[7] Omega project home page. http://www.ens.fr/omega/.

[8] PdfTeX project home page. http://www.tug.org/applications/pdftex/.

[9] MathML home page. http://www.w3.org/math/.

[10] *К. О. Тельников, П. З. Чеботаев.* LATEX. Издательская система для всех. Новосибирск: Сибирский хронограф, 1994.

[11] *И. А. Котельников, П. З. Чеботаев.* Издательская система LATEX 2_ε. Новосибирск: Сибирский хронограф, 1998.

[12] *Michel Goossens, Frank Mittelbach, and Alexander Samarin.* The LaTeX Companion. Reading, Massachusetts: Addison-Wesley, 1994.

Имеется перевод: *М. Гуссенс, Ф. Миттельбах, А. Самарин.* Путеводитель по пакету LaTeX и его расширению LaTeX 2_ε. М.: Мир, 1999.

[13] *Michel Goossens, Sebastian Rahtz, and Frank Mittelbach.* The LaTeX Graphics Companion: Illustrating Documents with TeX and Postscript. Reading, Massachusetts: Addison-Wesley, 1997.

Имеется перевод: *М. Гуссенс, С. Ратц, Ф. Миттельбах.* Путеводитель по пакету LaTeX и его графическим расширениям. М.: Мир, 2002.

[14] *Michel Goossens, Sebastian Rahtz, Eitan M. Gurari, Ross Moore, Robert S. Sutor.* LaTeX Web Companion: Integrating TeX, HTML, and XML. Reading, Massachusetts: Addison Wesley Professional, 1999.

Имеется перевод: *М. Гуссенс, С. Ратц, Э. Гурари, Р. Мур, Р. Сьютор.* Путеводитель по пакету LaTeX и его web-приложениям. М.: Мир, 2001.

[15] *George A. Gratzer.* First Steps in Latex. New York: Springer-Verlag, 1999.

Имеется перевод: *Г. Грэтцер.* Первые шаги в LaTeX. М.: Мир, 2000.

[16] *С. М. Львовский.* Набор и верстка в пакете LaTeX. 3-е изд., испр. и доп. М.: МЦНМО, 2003.

[17] *Keith Reikdahl.* Using EPS Graphics in LaTeX 2_ε Documents (распространяется в виде файла), 1996.

[18] *Oren Patashnik.* BibTeXing (распространяется в виде файла `btxdoc.tex`), 1988.

[19] *Pehong Chen and Michael A. Harrison.* Automating Index Preparation. Technical Report 87/347 (распространяется в виде файла `makeindx.tex`). Computer Science Division, University of California, Berkeley, 1987.

[20] *LaTeX3 Project Team.* LaTeX 2_ε Font Selection (распространяется в виде файла `fntguide.tex`), 1998.

[21] *Ю. Ярмола.* Компьютерные шрифты. СПб.: BHV — Санкт-Петербург, 1994.

[22] Portable Document Format Reference. Fourth Edition, Version 1.5. Adobe Systems Inc., 2003. http://partners.adobe.com/asn/tech/pdf/specifications.jsp.

[23] *Tomas Merz.* Web Publishing with Acrobat/PDF. Springer-Verlag, 1997.

Оглавление

Игорь Котельников
Платон Чеботаев



Редактор Н. Р. Тевс
Корректор Е. В. Панкратова
Художник И. А. Котельников

Уважаемый читатель!

Ваши замечания о содержании книги
просим направлять в издательство по адресу:

630060, г. Новосибирск, Зелёная Горка, 1,
НИЦ Сибирский хронограф
E-mail: sibirsky_khronograf@hotmail.com

или авторам книги по электронной почте
i.a.kotelnikov@inp.nsk.su

Подписано в печать с авторского оригинал-макета 10.03.2004. Формат 70 × 100/16. Печать офсетная. Усл. печ. л. 40. Тираж 4 000 экз. Заказ №15. Цена договорная.

Научно-издательский центр «Сибирский хронограф».
Отпечатано в типографии НИЦ «Сибирский хронограф».
630060, Новосибирск, Зелёная Горка, 1.
E-mail: sibirsky_khronograf@hotmail.com